Sun WorkShop 6 update 2 の新機能

Forte Developer 6 update 2
(Sun WorkShop 6 update 2)

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650-960-1300

Part No. 816-0883-01
2001 年 8 月 Revision A

原典： *What's New in Sun WorkShop 6 update 2*
Part No: 806-7982-10
Revision A

製品名の変更について

Sun は新しい開発製品戦略の一環として、Sun の開発ツール群の製品名を Sun WorkShop™ から Forte™ Developer に変更いたしました。製品自体の内容に変更はなく、従来通りの高品質をお届けいたします。

これまでの Sun の主力製品である基本プログラミングツールに、Forte Fusion™ や Forte™ for Java™ といった Forte 開発ツールの得意とする、マルチプラットフォームおよびビジネスアプリケーション実装の機能を盛り込むことで、より広範囲できめ細かな製品ラインが完成されました。

WorkShop 5.0 で使用されていた名称と、Forte Developer 6 で使用される新しい名称の対応については、以下の表をご覧ください。

旧名称	新名称
Sun Visual WorkShop™ C++	Forte™ C++ Enterprise Edition 6
Sun Visual WorkShop™ C++ Personal Edition	Forte™ C++ Personal Edition 6
Sun Performance WorkShop™ Fortran	Forte™ for High Performance Computing 6
Sun Performance WorkShop™ Fortran Personal Edition	Forte™ Fortran Desktop Edition 6
Sun WorkShop Professional™ C	Forte™ C 6
Sun WorkShop™ University Edition	Forte™ Developer University Edition 6

製品名の変更に加えて、次の 2 つの製品について大きな変更があります。

- Forte for High Performance Computing には Sun Performance WorkShop Fortran に含まれていたすべてのツール、および C++ コンパイラが含まれます。したがって、High Performance Computing のユーザーは開発用に 1 つの製品だけを購入すれば済むことになります。
- Forte Fortran Desktop Edition は以前の Sun Performance WorkShop Personal Edition と同じです。ただし、この製品に含まれる Fortran コンパイラでは、自動並列化されたコード、および明示的な指令に基づいた並列コードは生成できません。この機能は Forte for High Performance Computing に含まれる Fortran コンパイラでは使用できます。

Sun の開発製品を引き続きご利用いただきましてありがとうございます。今後もみなさまのご要望にお応えする製品をお届けできるよう努力してまいります。

目次

はじめに

『Sun WorkShop 6 update 2 の新機能』では、Sun WorkShop™ 6 update 2 のコンパイラとツールの新しい機能 (および Sun WorkShop 6 と Sun WorkShop 6 update 1 の各リリースの新しい機能) 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` *filename* 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。	`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)	#

サポートしているプラットフォーム

この Sun WorkShop™ リリースでは、Solaris™ SPARC™ プラットフォーム版と Solaris™ Intel プラットフォーム版をオペレーティング環境とするバージョン 2.6、7、および 8 をサポートしています。

Sun WorkShop の開発ツールとマニュアルページへのアクセス

Sun WorkShop の製品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ていません。SunWorkShop のコンパイラとツールにアクセスするには、PATH 環境変数に SunWorkshop コンポーネントディレクトリを必要とします。SunWorkshop マニュアルページにアクセスするには、PATH 環境変数に SunWorkshop マニュアルページが必要です。

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、『Sun WorkShop 6 update 2 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 WorkShop 6 update 2 製品が /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。 Sun WorkShop 製品が /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。

Sun WorkShop 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 Sun WorkShop 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 Sun WorkShop 開発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 Sun WorkShop の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

Sun WorkShop マニュアルページへのアクセス方法

Sun WorkShop 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、workshop マニュアルページを表示します。

```
% man workshop
```

2. 出力された場合、内容を確認します。

   workshop(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 Sun WorkShop 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
```

Sun WorkShop マニュアルへのアクセス

Sun WorkShop の製品マニュアルには、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。

  Netscape™ Communicator 4.0 または互換性がある Netscape バージョンのブラウザで次のファイルにポイントします。

  ```
  /opt/SUNWspro/docs/ja/index.html
  ```

  製品ソフトウェアが /opt ディレクトリにインストールされていない場合は、システム上でこのディレクトリに相当するパスをシステム管理者に問い合わせてください。

- マニュアルは、docs.sun.com の Web サイトで入手できます。

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読んだり、印刷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。

関連マニュアル

次の表では、docs.sun.com の Web サイトで利用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
数値計算ガイド	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。
Solaris 8 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 8 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 8 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。

Sun のマニュアルの注文

製品マニュアルは `docs.sun.com` Web サイトまたは Fatbrain.com インターネットブックストアを通じて米国 Sun Microsystems,Inc. に直接注文できます。Fatbrain.com の Sun Documentation Center へは次の URL でアクセスできます。

`http://www.fatbrain.com/documentation/sun`

ご意見の送付先

米国 Sun Microsystems, Inc. では、マニュアルの向上に力を注いでおり、ユーザーのご意見やご提案をお待ちしております。ご意見などがありましたら、次のアドレスまで電子メールをお送りください。

`docfeedback@sun.com`

第1章

Sun WorkShop 6 update 2 の新機能の紹介

この章では、Sun WorkShop™ 6 update 2 のコンパイラとツールの新しい機能について説明します。このリリースは、主にパフォーマンス向上、非標準 C++ ソースコードの受け入れ、および拡充した OpenMP サポートにねらいを定めています。

この章に含まれる節は次のとおりです。

マニュアル

この章で説明する Sun WorkShopマニュアルにアクセスするには、このマニュアルの xv ページの「Sun WorkShop マニュアルへのアクセス」を参照してください。

このリリースで新たに再編成されたマニュアルの説明については、『Sun WorkShop 6 update 2 マニュアルの概要』の第 2 章を参照してください。
Sun WorkShop 6 update 2 に関する本のタイトルと説明については、『Sun WorkShop 6 update 2 マニュアルの概要』の第 3 章を参照してください。

C コンパイラ

次の表に一覧表示された新しい各機能の詳細については、『C ユーザーズガイド』を参照してください (各オプションには本の索引に一覧があります)。

表 1-1　C コンパイラの新機能

機能	オプションまたはマクロ	説明
OpenMP	`-xopenmp`	OpenMP C および C++ Application Program Interface Version 1.0 - 1998 年 10 月に記載したプラグマを認識できます。この仕様は `http://www.openmp.org` で入手できます。
C99 の部分的サポート	`-xc99`	コンパイラで、C99 標準の機能の一部に関する構文や意味を受け入れることができます。
C99 機能の `lint` サポート	`-Xc99`	`lint` を使用可能にし、サポートしている C99 機能がコードで正しく呼び出されるかどうかを確認します。
型に基づいた別名解析の `lint` サポート	`-Xalias_level`	`lint` を使用して、コンパイル時に指定する予定の型に基づいた別名解析にコードがどの程度密接に準拠しているかを確認します。

表 1-1　C コンパイラの新機能 (続き)

機能	オプションまたはマクロ	説明
内部手続きの最適化	`-xipo`	内部手続き解析パスを呼び出すことで全プログラムの最適化を実行します。`-xcrossfile` とは違って、`-xipo` リンク手順におけるすべてのオブジェクトファイル間の最適化を実行します。しかも最適化の対象はコンパイルコマンドのソースファイルに限定されません。
コードセットを独立してサポート	`-xcsi`	C コンパイラで ISO C ソース文字コード要件に準拠しないロケールで作成されたソースコードを受け入れられるようにします。これらのロケールには、`ja_JP.PCK` が含まれます。
Char 型が符号付きかどうかの識別	`-xchar`	このオプションは、char を符号なしで定義するシステムからコードを移行する場合にのみ使用します。char が符号付きかどうかがコードに無関係である場合はこのオプションを使用しないでください。符号付きまたは符号なしの char を明示的に指定するためにこのコードを書き直す必要はありません。

C++ コンパイラ

次の表には、C++ コンパイラ (5.3) の Sun WorkShop 6 update 2 リリースで利用できる新しい機能を一覧表示します。一部の機能については、表の次の節でさらに詳しく説明しています。これらの機能についてはすべて『C++ ユーザーズガイド』 で説明しています。

表 1-2　C++ コンパイラの新しい機能

機能	オプション	説明
Tools.h++ ライブラリの標準 iostream バージョン	`-library=rwtools7_std`	標準 iostream で動作する Tools.h++ ライブラリバージョンを利用できるようにします。
共有`libCstd`		`libCstd` を動的にリンクできます。
共有 `libiostream`		`libiostream` を動的にリンクできます。
パフォーマンスの向上		Sun WorkShop 6 update 1 C++ コンパイラ (5.2) のリリース後にパフォーマンスの向上がいくつか行われました。
古い警告の管理を強化	`-features=[no%]transitions`	一部の古い警告はエラーになる可能性があります。
非標準ソースコードの受け入れ	`-features=[no%]extensions`	この新しいオプションを使用すると、他の C++ コンパイラによって一般に受け入れられている非標準コードをコンパイルできます。
新しい引数により `-xinline` オプションが復元済み	`-xinline`	`-xinline` オプションが復元されました。さらに、新しい `%auto` 引数がとられ、これによって -xO4 またはそれ以上の最適化レベルで自動インライン化が実現できます。

表 1-2　C++ コンパイラの新しい機能 (続き)

機能	オプション	説明
さらに効果的な標準ライブラリ呼び出しの最適化	`-xbuiltin`	この新しい -xbuiltin オプションでは、できるだけ多数の組み込み標準関数を認識するようにコンパイラに指示します。
拡張 `-fast` オプション	`-fast`	`-fast` オプションは `-xbuiltin=%all` オプションを含むように拡張されました。
内部手続きの最適化	`-xipo`	この新しい -xipo オプションは内部手続き解析パスを呼び出すことで全プログラムの最適化を実行します。
ライセンス情報をコンパイルコマンドの一部として要求	`-xlicinfo`	このオプションによって同じコマンドでライセンス情報の取得とコンパイルを同時にできます。
変数引数プリプロセッサ関数に似たマクロ		C++ コンパイラは C99 標準の定義に従って引数の変数をとることのできるプリプロセッサ関数マクロをサポートします。
`enum` 宣言の後にくるコンマがエラーとして扱われなくなった		コンパイラが `enum` 宣言の後にくるコンマを受け入れ、警告を出すようになりました。
非局所静的オブジェクト初期設定子の取り扱いに関する新しいオプション	`-features=[no%]split_init`	このオプションは、すべての初期設定子を 1 つの関数に入れるか別々の関数に入れる (デフォルト) かを指定する場合に使用します。

Tools.h++ ライブラリの標準 iostream バージョン

C++ コンパイラのこのリリースには 2 つの Tools.h++ ライブラリバージョンがあります。

- 典型的な iostream バージョン。このバージョンの Tools.h++ ライブラリは、このコンパイラの以前のバージョンでリリースした Tools.h++ ライブラリと互換性があります。
  - Tools.h++ の典型的な iostream バージョンを標準モード (デフォルトモード) で使用するには、`-library=rwtools7,iostream` オプションを使用します。
  - Tools.h++ の典型的な iostream バージョンを互換モード (`-compat[=4]`) で使用するには、`-library=rwtools7` オプションを使用します。
- 標準 iostream バージョン。 このバージョンの Tools.h++ ライブラリは、Tools.h++ の典型的な iostream バージョンと互換性があります。このバージョンは、標準モードでのみ使用でき、互換性モードでは使用できません。(-compat[=4]).

  ライブラリの標準 iostream バージョンを使用するには、`-library=rwtools7_std` オプションを使用してください。

共有`libCstd`

Sun WorkShop 6 update 2 C++ コンパイラ (5.3) には、`libCstd` ライブラリの共有バージョンが含まれます。

`libCstd` の共有バージョンを使用する手順は次のとおりです。

1. スーパーユーザとして、次のシンボリックリンクを手動で作成します。

```
example% ln -s /usr/lib/libCstd.so.1 \
  /opt/SUNWSpro/lib/libCstd.so
example% ln -s /usr/lib/libCstd.so.1 \
  /opt/SUNWspro/lib/v8plus/libCstd.so
example% ln -s /usr/lib/sparcv9/libCstd.so.1 \
  /opt/SUNWSpro/lib/v9/libCstd.so
```

プロセッサアーキテクチャーが Intel 32 ビットの場合は、最後の 2 つのリンクは必要ありません。

注 - コンパイラが /opt/SUNWSpro ディレクトリにインストールされていない場合は、システム上でこのディレクトリに相当するパスをシステム管理者に問い合わせてください。

---

2. リンクをテストします。

   /opt/SUNWSpro/bin/CC で任意のプログラムをコンパイルしたら、ldd a.out コマンドを入力します。/usr/lib/libCstd.so.1 に依存関係があれば出力されます。

   これらのシンボリックリンクを作成すると、libCstd はデフォルトにより動的にリンクされます。

   libCstd を静的にリンクするには、-staticlib=Cstd オプションを使用します。

   共有 libCstd ライブラリとリンクしたオブジェクトファイルを配布する予定の場合は、次のいずれかを行う必要があります。

   - 製品の最新の SUNWlibC パッチを配布します。

     または

   - http://sunsolve.sun.com など、Sun の Web サイトから最新の SUNWlibC パッチをダウンロードするよう顧客に要求します。

共有 libiostream

Sun WorkShop 6 update 2 C++ コンパイラ (5.3) には、典型的な iostream ライブラリ libiostream の共有バージョンが含まれます。

libiostream の共有バージョンを使用する手順は次のとおりです。

1. スーパーユーザーとして、次のシンボリックリンクを手動で作成します。

```
example% ln -s /usr/lib/libiostream.so.1 \
  /opt/SUNWspro/lib/libiostream.so
example% ln -s /usr/lib/sparcv9/libiostream.so.1 \
  /opt/SUNWSpro/lib/v9/libiostream.so
```

   プロセッサアーキテクチャーが Intel 32 ビットの場合は、最後のリンクは必要ありません。

注 - コンパイラが /opt/SUNWSpro ディレクトリにインストールされていない場合は、システム上でこのディレクトリに相当するパスをシステム管理者に問い合わせてください。

---

2. リンクをテストします。

   オプション -library=iostream を使用して、任意のプログラムを /opt/SUNWSpro/bin/CC でコンパイルします。次に、ldd a.out コマンドを入力します。/usr/lib/libiostream.so.1 に依存関係があれば出力されます。

   これらのシンボリックリンクを作成すると、コンパイラでは -library=iostream を使用するたびにデフォルトにより libiostream が動的にリンクされます。

   このライブラリを静的にリンクするには、-library=iostream -staticlib=iostream オプションを使用します。

   共有 libiostream ライブラリとリンクしたオブジェクトファイルを配布する予定の場合は、次のいずれかを行う必要があります。

   - 製品の最新の SUNWlibC パッチを配布します。

     または

   - http://sunsolve.sun.com など、Sun の Web サイトから最新の SUNWlibC パッチをダウンロードするよう顧客に要求します。

パフォーマンスの向上

Sun WorkShop 6 update 2 C++ (5.3) コンパイラでは、(5.2 C++ コンパイラと比較すると) 次の点でパフォーマンスが向上しています。

- 特にテンプレートを多用している同じディレクトリ内での並列コンパイルでコンパイル時間が著しく短縮されています。
- 標準ライブラリ、特に iostream と文字列クラスで効果的なパフォーマンスを提供します。文字列クラスの向上は、-xarch=v8plus オプションまたは -xarch=v9 オプションを選択した場合の特殊コード生成に依存しています。

- インライン関数の生成をさらに効果的に管理します。以前はコンパイラの複雑さに対する許容範囲を超えると考えられてきた一部の関数は現在はインライン化されています。デフォルトのインライン化は、低い最適化レベルでコンパイル時間を短縮し、かつ高い最適化レベルで実行時間を高速化するように調整されました。

古い警告の管理を強化

互換モード (`-compat [=4]`) では、コンパイラは移行エラーが発生してもそれを報告しません。`+w` または `+w2` を指定すると、コンパイラはこれらのエラーを発見すると警告を出します。

標準モードでは、移行エラーに関する警告が出されます。

互換モードまたは標準モードで新しい `-features=no%transitions` オプションを指定すると、コンパイラは警告ではなくエラーメッセージを出します。

新しい `-features=no%transitions` オプションで制御される言語構造は次のとおりです。

- テンプレートを使用した後にテンプレートを再定義する
- テンプレートの定義で必要とされる場合は `typename` 指示を削除する
- `int` 型を暗黙に宣言する

詳細は、『C++ ユーザーズガイド』を参照してください。

非標準ソースコードの受け入れ

新しい `-features=extensions` オプションを使用すると、他の C++ コンパイラによって一般に受け入れられている非標準コードをコンパイルできます。

このオプションは、無効なコードをコンパイルする必要があるとき (コードを有効にするための修正が許可されていないとき) に使用できます。

注 - サポートされている無効なコードのインスタンスをそれぞれすべてのコンパイラで受け入れられる有効なコードに変更することができます。コードを無効にする許可を持っている場合は、このオプションを使用しないで無効にしてください。`-features=extensions` オプションを使用すると、無効なコードが永続化されますが、これらのコードは一部のコンパイラで拒否されます。

`-features=extensions` をコンパイルコマンドに追加すると、コンパイラで次の言語拡張機能がサポートされます。

- 仮想関数の上書きは、上書きする関数よりも制限が緩和される場合があります。
- `enum` 型と変数の順方向宣言が可能です。
- 不完全な `enum` 型はフォワード宣言とみなされます。
- `enum` 名をスコープ修飾子として使用できます。
- 無名 `struct` 宣言が可能です。
- 無名クラスインスタンスのアドレスの引き渡しが可能です。
- 静的名前空間スコープ関数をクラスのフレンドとして使用できます。
- 関数名に事前定義された `__func__` シンボルを使用できます。

詳細は、『C++ ユーザーズガイド』 を参照してください。

Fortran コンパイラ

Sun WorkShop 6 update 2 では、Fortran 95 (`f95`) コンパイラと Fortran 77 (`f77`) コンパイラの両方を提供しています。下記の表では、Sun WorkShop 6 update 2 リリースの Fortran 95 コンパイラの新しい機能を一覧表示しています。Fortran 77 については新しい機能はリリースされていません。詳細については、『Fortran ユーザーズガイド』を参照してください。

表 1-3　Fortran 95 コンパイラの新機能

機能	説明
VAX 構造	f77 からのプログラムの移行をサポートするために、f95 では Fortran 95 の派生型の原型である VAX Fortran の STRUCTURE と UNION の各文を受け入れます。『Fortran ユーザーズガイド』の付録 C を参照してください。
拡張 ALLOCATABLE 属性	Fortran 95 標準化機構では、ALLOCATABLE 属性で使用できるデータ要素を拡張しました。以前はその使用がローカル配列に限定されていた ALLOCATABLE 属性が、構造の配列成分、ダミー配列、および配列関数結果で使用できるようになりました。『Fortran ユーザーズガイド』の付録 C を参照してください。
VALUE 属性	f95 は、Fortran 2000 標準で提案している機能である VALUE 属性を認識します。サブプログラムのダミー引数を VALUE で宣言すると、実際の引数が参照によってではなく値によって引き渡されることを示します。『Fortran ユーザーズガイド』の付録 C を参照してください。
ストリーム入出力	Fortran 2000 で提案している別の機能に、1 で始まる正の整数でアドレス指定できる連続バイトシーケンスとしてデータファイルを扱う新しいストリーム入出力スキーマがあります。ストリーム入出力を使用可能にするには、ファイルを ACCESS='STREAM' で宣言します。ファイルを POS=*integer_expression* 指示子を使用して READ 文または WRITE 文で位置付けます。『Fortran ユーザーズガイド』の付録 C を参照してください。
大域プログラムチェック	-Xlist オプションによって呼び出される f95 の大域プログラムチェック機能は、f77 と同様、-Xlistc、-Xlisth、 -Xlists、-Xlistv*n*、および -Xlistw[*n*] の各サブオプションを含むようになりました。『Fortran ユーザーズガイド』の第 3 章を参照してください。

表 1-3　Fortran 95 コンパイラの新機能 (続き)

機能	説明
Fortran ライブラリインタフェース	f95 では、Fortran ライブラリに対して正しいデータ型を宣言するためにインクルードファイルの system.inc を認識します。戻り値が確実に正しく入力されるように、非組み込み Fortran ライブラリルーチンを参照するすべてのルーチンに INCLUDE 'system.inc' 文を含めてください。
内部手続きの最適化	新しい -xipo オプションを指定すると、コンパイラはプログラム全体の最適化を実行します (サブプログラムのインライン化)。-xipo オプションは -xcrossfile オプションと異なり、すべてのファイルが 1 つのコンパイルコマンドに存在しなくても構いません。-xipo は、大きい複数ファイルのアプリケーションをコンパイルまたはリンクするときに便利です。『Fortran ユーザーズガイド』の第 3 章を参照してください。
OpenMP 2.0 の拡張機能	f95 は、OpenMP 2.0 Fortran API をサポートするようになりました。拡張機能には、WORKSHARE、配列を対象とする REDUCTION、変数を対象とする THREADPRIVATE、および SINGLE 指示を対象とする COPYPRIVATE があります。『Fortran ユーザーズガイド』の付録 E および http://www.openmp.org を参照してください。
OpenMP ライブラリインタフェース	新しいコンパイラは、 OpenMP Fortran ライブラリルーチンへのインタフェース定義するためにインクルードファイルとインタフェースモジュールを備えています。確実に正しい型が宣言されるように OpenMP ライブラリを参照するすべてのプログラムユニットに、INCLUDE 'omp_lib.h' または USE omp_lib のいずれかの文を指定します。『Fortran ユーザーズガイド』の付録 E を参照してください。

dbx

dbx の新しい機能の一部は、表の次の節でさらに詳しく説明しています。

表 1-4　dbx の新機能

機能	説明
実行時検査での 8 メガバイトの制限の廃除	実行時検査での 8 メガバイトの制限は UltraSPARC プロセッサ搭載ハードウェアには適用されなくなりました。このプロセッサでは、dbx に分岐を使用する代わりにトラップハンドラを呼び出す機能が備わっています。
一致しないコアファイルのデバッグ	「一致しない」コアファイルのデバッグに関する事前サポート (たとえば、Solaris オペレーティング環境の異なるバージョンまたはパッチレベルのシステムで生成されるコアファイル) を利用できるようになりました。
step to コマンド	新しい step to コマンドは、現在のソースコード行の指定関数にステップインしようとします。
gdb コマンドのサポート	gbd コマンドでは、gdb コマンドセットをサポートします。
collector コマンドに対する変更	collector pause コマンドと collector resume コマンドが新たに加わりました。一方、collector enable_once コマンドが削除されました。collector store コマンドに実験グループが含まれるように拡張されました。詳細は、『dbx コマンドによるデバック』の付録 C の「collector コマンド」および collector(1) マニュアルページを参照してください。
Intel プラットフォームでの Fortran 組み込み関数のサポートを削除	Intel プラットフォームでの Fortran 組み込み関数のサポートが削除されました。

実行時検査での 8 メガバイトの制限の廃除

実行時検査での 8 メガバイトの制限は UltraSPARC™ プロセッサ搭載ハードウェアではなくなりました。このプロセッサでは、dbx は分岐を使用する代わりにトラップハンドラを呼び出すことができます。トラップハンドラへの制御の移行には、最高で 10

倍の時間がかかりますが、8 メガバイトの制限はありません。トラップは、ハードウェアが UltraSPARC プロセッサに基づいている限り、必要に応じて自動的に使用されます。ハードウェアを確認するには、システムコマンド `isalist` を使用し、実行結果に文字列 `sparcv8plus` が入っていることを確認してください。

dbx 環境変数 `rtc_use_traps` が削除されました。

一致しないコアファイルのデバッグ

プログラムがどこでクラッシュしているかを確認するには、プログラムがクラッシュしたときにメモリーイメージであるコアファイルを調べるという方法があります。コアをダンプしたプログラムが任意の共有ライブラリと動的にリンクされた場合は、コアファイルが作成されたのと同じオペレーティング環境でコアファイルをデバッグするのが最適です。しかし、それが必ずしも可能あるいは便利であるとは限りません。現在、dbx による「不一致の」コアファイル (Solaris オペレーティング環境の異なるバージョンまたはパッチレベルのシステムで生成されるコアファイルなど) のデバッグに対するサポートは制限されています。

不一致のコアファイルをデバッグする場合、次の 2 つの問題がライブラリで生じる可能性があります。

- コアホスト上のプログラムで使用される共有ライブラリは、dbx ホストで使用されるライブラリとは異なります。ライブラリに関連する正確なスタックトレースを入手するには、これらのオリジナルライブラリを dbx ホスト上で使用できるようにする必要があります。
- dbx は `/usr/lib` ディレクトリ内のシステムライブラリを使用して、システム上の実行時リンカーとスレッドライブラリの実装状態の詳細を分かりやすくします。場合によっては、これらのシステムライブラリをコアホストから提供し、dbx で実行時リンカーやスレッドのデータ構造を認識できるようにする必要があります。

これらのライブラリに関する問題を取り除き、dbx で不一致のコアファイルをデバッグする方法については、『dbx コマンドによるデバッグ』の第 2 章の「一致しないコアファイルのデバッグ」を参照してください。

step to コマンド

step to コマンドは、現在のソースコード行の指定関数にステップインしようとします。関数が指定されていない場合、このコマンドは、現在のソースコード行のアセンブリコードで決められたとおり、呼び出された最後の関数にステップインしようとします。

この関数呼び出しは、条件付き分岐であるために行われない可能性があります。呼び出しが行われない場合、または現在のソースコード行に関数呼び出しがない場合は、step to コマンドは現在のソースコード行をステップオーバーします。

step to コマンドの詳細については、『dbx コマンドによるデバック』の付録 C の「step コマンド」を参照してください。

gdb コマンドのサポート

gdb コマンドでは、gdb コマンドセットをサポートします。gdb on コマンドを使用すると、dbx が gdb コマンドを認識して受け入れる gdb コマンドモードに入ります。gdb コマンドモードを終了し、dbx コマンドモードに戻るには、gdb off コマンドを使用します。dbx コマンドは gdb コマンドモードでは受け入れられません。また、gdb コマンドは dbx コマンドモードでは使用できません。すべてのデバッグの設定値 (ブレークポイントなど) は、異なるコマンドモード間で保持されています。

次の gdb コマンドはこのリリースでサポートしていません。

- command
- define
- handle
- hbreak
- interrupt
- maintenance
- printf
- rbreak
- return
- signal
- tcatch
- until

Sun WorkShop TeamWare

表 1-5　Sun WorkShop TeamWare の新機能

機能	説明
ヘルプメニューからアクセスできる Quick Tour	Sun WorkShop TeamWare の基本的なモデルと機能の詳しい概要を提供する対話型の Quick Tour は、どの Sun WorkShop TeamWare ウィンドウのヘルプメニューからもアクセスできるようになりました (ヘルプ ➤ TeamWare Quick Tour を選択)。

Sun Performance Library

Sun Performance Library™ は、線形代数の問題やその他数値を多用する問題を解決するための最適化された一連の高速の数学サブルーチンです。Sun Performance Library は、Netlib (`http://www.netlib.org`) から入手できる一連のパブリックドメインアプリケーションに基づいています。これらのルーチンは機能が拡張され、

Sun Performance Library としてバンドルされています。

表 1-6　Sun Performance Library の新機能

機能	説明
パフォーマンスの向上	パフォーマンスは次のルーチンについて向上しました。 • LU 因数分解ルーチン (GETRF) • ロングベクトル複素数および倍複素数 1D FFT ルーチン (ここで、ロングベクトルは より大きい131072 より大きいとして定義されています) • 選択された BLAS ルーチン • スパースソルバー数値因数分解ルーチンおよび解決ルーチン
並列化されたその他のルーチン	ロングベクトル複素数および倍複素数の 1D FFT ルーチンが並列化されました。
スパースソルバーの拡張機能	スパースシステムによっては、メモリの使用量を削減し、実行時間を短縮するために、スパースソルバーにさらにスパースマトリックス順序付け手法が追加されました。

LINPACK は Sun Performance Library の次回のバージョンから削除されます

Sun Performance Library の Sun WorkShop 6 update 2 の次のリリースでは、LINPACK はライブラリに含まれません。LAPACK バージョン 3.0 は LINPACK および以前の LAPACK のすべてのバージョンに取って代わります。LINPACK ルーチンを呼び出す従来のユーザコードを LAPACK ルーチンを使用するように修正できない場合でも、LINPACK のパブリックドメインバージョンは引き続き Netlib から入手できます。

表 1-7　次回のリリースで削除される LINPACK ルーチン

CCHDC	DCHDC	SCHDC	ZCHDC	CCHDD	DCHDD	SCHDD	ZCHDD
CCHEX	DCHEX	SCHEX	ZCHEX	CCHUD	DCHUD	SCHUD	ZCHUD
CGBCO	DGBCO	SGBCO	ZGBCO	CGBDI	DGBDI	SGBDI	ZGBDI
CGBFA	DGBFA	SGBFA	ZGBFA	CGBSL	DGBSL	SGBSL	ZGBSL
CGECO	DGECO	SGECO	ZGECO	CGEDI	DGEDI	SGEDI	ZGEDI

表 1-7　次回のリリースで削除される LINPACK ルーチン (続き)

CGEFA	DGEFA	SGEFA	ZGEFA	CGESL	DGESL	SGESL	ZGESL
CGTSL	DGTSL	SGTSL	ZGTSL	CHICO	ZHICO	CHIDI	ZHIDI
CHIFA	ZHIFA	CHISL	ZHISL	CHPCO	ZHPCO	CHPDI	ZHPDI
CHPFA	ZHPFA	CHPSL	ZHPSL	CPBCO	DPBCO	SPBCO	ZPBCO
CPBDI	DPBDI	SPBDI	ZPBDI	CPBFA	DPBFA	SPBFA	ZPBFA
CPBSL	DPBSL	SPBSL	ZPBSL	CPOCO	DPOCO	SPOCO	ZPOCO
CPODI	DPODI	SPODI	ZPODI	CPOFA	DPOFA	SPOFA	ZPOFA
CPOSL	DPOSL	SPOSL	ZPOSL	CPPCO	DPPCO	SPPCO	ZPPCO
CPPDI	DPPDI	SPPDI	ZPPDI	CPPFA	DPPFA	SPPFA	ZPPFA
CPPSL	DPPSL	SPPSL	ZPPSL	CPTSL	DPTSL	SPTSL	ZPTSL
CQRDC	DQRDC	SQRDC	ZQRDC	CQRSL	DQRSL	SQRSL	ZQRSL
CSICO	DSICO	SSICO	ZSICO	CSIDI	DSIDI	SSIDI	ZSIDI
CSIFA	DSIFA	SSIFA	ZSIFA	CSISL	DSISL	SSISL	ZSISL
CSPCO	DSPCO	SSPCO	ZSPCO	CSPDI	DSPDI	SSPDI	ZSPDI
CSPFA	DSPFA	SSPFA	ZSPFA	CSPSL	DSPSL	SSPSL	ZSPSL
CSVDC	DSVDC	SSVDC	ZSVDC	CTRCO	DTRCO	STRCO	ZTRCO
CTRDI	DTRDI	STRDI	ZTRDI	CTRSL	DTRSL	STRSL	ZTRSL

標本アナライザ

次の表で説明している新機能の詳しい説明については、『プログラムのパフォーマンス解析』および analyzer(1)、collect(1)、collector(1)、dbx(1)、er_print(1)、er_src(1) および libcollector(3) の各マニュアルページを参照してください。

表 1-8　標本アナライザの新機能

機能	説明
デュアルハードウェアカウンタのオーバーフローの実験	2 つのハードウェアカウンタのデータは、カウンタが異なるレジスタを使用している場合に限り、同じ実験で収集できます。
コンパイラの注釈	コンパイラの注釈は、注釈付きのソースコードに表示されます。標本コレクタや標本アナライザを実行しなくてもコンパイラの注釈を表示するために新しいスタンドアロンのソースブラウザ、er_src を利用できます。
データ収集の一時停止および再開	データ収集は、実行の過程で一時停止や再開を行うことができます。この機能は、シグナルを使用した collect コマンド、pause および resume サブコマンドを使用した dbx collector コマンドでも利用できますし、libcollector API からも利用できます。
MPI の実験名	MPI の実験のデフォルト名には、MPI のランクが含まれます。
実験名の標準	実験名は、末尾が必ず .er で、実験グループ名は末尾が必ず .erg でなければなりません。
実験グループのサポート	dbx では dbx collector store group サブコマンドを使って実験グループを使用できます。

表 1-8　標本アナライザの新機能 (続き)

機能	説明
デフォルトファイル	標本アナライザおよび er_print コマンドは、メトリックや他の機能のデフォルト値を記述しておくことができる .er.rc という名前のデフォルトファイルを読み取ります。
ハードウェアカウンタメトリック名の変更	一部のハードウェアカウンタメトリックの名前が変更されました。
dbx コマンドの廃止	collector enable_once、collector close、および collector quit コマンドは現在サポートされていません。

第2章

Sun WorkShop 6 update 1 の新機能の紹介

この章では、Sun WorkShop™ 6 update 1 のコンパイラとツールの新しい機能について説明します。このリリースは、主に UltraSPARC™ III プロセッサにおけるパフォーマンス向上にねらいを定めています。

この章は更新された Sun WorkShop コンポーネントごとに節に分かれており、新機能の概要が表にまとめられています。一部の機能については、表の次の節でさらに詳しく説明しています。

この章に含まれる節は次のとおりです。

C コンパイラ

表 2-1 は、C コンパイラの Sun WorkShop 6 update 1 リリースで使用できる新機能を示しています。これらの機能の一部は、表の次の節でさらに詳しく説明しています。

表 2-1　C コンパイラの新機能

機能	オプションまたはマクロ	説明
UltraSPARC III プロセッサのサポート	`-xtarget` `-xchip`	新リリースの `-xtarget` および `-xchip` オプションには、`ultra3` を指定できるようになりました。UltraSPARC III で最良のパフォーマンスを得るために推奨されるフラグについては、次の節の説明を参照してください。
型に基づいた解析による最適化	`-xalias_level`	新リリースの C コンパイラでは、型に基づいた別名解析と最適化を行うための、オプションとプラグマが使用できます。
新しいプラグマによる数学ルーチンパフォーマンスの向上	`__MATHERR_ERRNO_DONTCARE`	新リリースの `cc -fast` コマンドの展開には、マクロ `__MATHERR_ERRNO_DONTCARE` が含まれています。このマクロにより Solaris 8 オペレーティング環境の `math.h` は、`math.h` でプロトタイプ化されている数学ルーチンの一部に対して、パフォーマンスに関連したプラグマを表明します。
標準のライブラリ関数のインライン化	`-xbuiltin`	このオプションを使用すると、コンパイラが有益であると判断した場合に、標準のライブラリ関数の組み込み関数を代用、またはインライン化することにより、生成されるコードのパフォーマンスが向上します。

表 2-1　C コンパイラの新機能 (続き)

機能	オプションまたはマクロ	説明
3 文字表記認識の有効化と無効化	`-xtrigraphs`	新しい `-xtrigraphs` オプションでは、3 文字表記変換を行うかどうかを指定できます。
先読み潜在期間の指定子	`-xprefetch=...,latx:`*factor*	この新しいサブオプションによって、データを先読みするためにコンパイラが生成する命令と、その後使用するロードまたはストア命令のデータがどれだけ離れているかを時間的に指定できます。
`-I-` オプションによるデフォルト検索パスの無効化	`-I-`	新しい `-I-` オプションを使用すると、コンパイラがインクルードファイルの検索時に使用するアルゴリズムを細かく制御できます。

UltraSPARC III プロセッサのサポート

UltraSPARC III プロセッサ上でプログラムをコンパイルして実行する場合に最良のパフォーマンスを得るには、以下のオプションを使用してコンパイルしてください。

```
-fast -xcrossfile -xprofile={collect:|use:}
```

- クロスコンパイルを行う (UltraSPARC III 以外のプラットフォームでコンパイルするが、UltraSPARC III システムで動作するようにオブジェクトバイナリを生成する) には、正しいキャッシュサイズと最適化方針が設定されるように以下のオプションを追加してください。

  ```
  -xtarget=ultra3 -xarch={v8plusb|v9b}
  ```

- 32 ビットコードの生成には `-xarch=v8plusb`、64 ビットコードの生成には `v9b` を指定してください。非常に大きなデータファイルにアクセスするプログラムの場合、64 ビットコードの方が優れたパフォーマンスを得られます。しかし、`-xarch=v9b` は 64 ビットコードを生成しなければならない場合だけに使用することをお勧めします。状況によっては、`-xarch=v9b` を使用したためにパフォーマンスが低下することがあります。

- `-fast` オプションは、実行速度を重視する最適化を有効にし、最適化レベルを `-xO5` に引き上げます。`-fast` を使用すると、可能な場合により高速な算術演算を代用できるようになるため、`-fsimple=2` を使用します。しかしこの場合、多少の誤りが発生する可能性があることに注意してください。
- `-xcrossfile` が指定されると、コンパイラはコマンド行に指定されたすべてのソースファイルに最適化 (中間手続き的な最適化を含む) を適用します。
- `-xprofile={collect:|use:}` は、プログラムパフォーマンスのプロファイルを有効にします。プロファイルによりコンパイラは、コードの中で最も頻繁に実行されているセクションを特定し、最良の効果を得るための局所的な最適化を行います。

注 - `-xarch={v8plusb|v9b}` により UltraSPARC III プラットフォーム向けにコンパイルされたプログラムは、UltraSPARC III 以外のプラットフォームでは動作しません。UltraSPARC I、UltraSPARC II、および UltraSPARC III の各プラットフォームで互換性を維持したまま動作するプログラムをコンパイルするには、`-xarch={v8plusa|v9a}` を使用してください。

型に基づいた解析による最適化

新しい C コンパイラコマンド `-xalias_level` およびいくつかの新しいプラグマを使用して、コンパイラに型に基づいた別名解析および最適化を行わせることができます。これらの拡張機能は、C プログラム内でのポインタの使用方法に関する型に基づいた情報を指定するために使用してください。C コンパイラは、この情報を使用してプログラム内のポインタに基づいたメモリー参照に対して、別名の明確化という極めて優れた処理を行います。

この新しいコンパイラコマンドの詳細は、`http://docs.sun.com` で入手できる『C ユーザーズガイドの追補』(Forte™ Developer 6 update 1/Sun WorkShop 6 update 1 Collection 内) を参照してください。

新しいプラグマによる数学ルーチンパフォーマンスの向上

新リリースでは、-fast オプションを発行するとマクロ __MATHERR_ERRNO_DONTCARE が定義されます。このマクロにより Solaris 8 オペレーティング環境の math.h は、<math.h> でプロトタイプ化されている数学ルーチンの一部に対して次のようなパフォーマンスに関連したプラグマを表明するようになります。

- #pragma does_not_read_global_data
- #pragma does_not_write_global_data
- #pragma no_side_effect

matherr(3M) のマニュアルページに説明されているように、例外的にコードがエラーの戻り値に依存する場合は、-fast オプションの後に -U__MATHERR_ERRNO_DONTCARE マクロを発行してマクロを無効にする必要があります。

標準のライブラリ関数のインライン化

標準のライブラリ関数を呼び出すコードの最適化レベルを向上させるには、-xbuiltin[=%all|%none] コマンドを使用してください。標準のライブラリ関数の多く (math.h や stdio.h などで定義されているものなど) は、一般にさまざまなプログラムによって使用されます。このコマンドが使用されると、コンパイラはシステム関数の代わりに組み込み純粋仮想関数を使用してパフォーマンスを向上させることができます。-xbuiltin コマンドの構文は次のとおりです。

- -xbuiltin=%all
- -xbuiltin=%none

このコマンドの第一のデフォルトは -xbuiltin=%none です。これは、標準ライブラリのどの関数もインライン化されないことを意味します。第一のデフォルトは、-xbuiltin が指定されなかった場合に適用されます。

第二のデフォルトは -xbuiltin=%all です。これは、最適化による改善が見込まれる場合にコンパイラが標準ライブラリ関数をインライン化することを意味しています。第二のデフォルトは、-xbuiltin が引数なしで指定された場合に適用されます。

-fast オプションを指定してコンパイルすると、-xbuiltin は %all に設定されます。

3 文字表記認識の有効化と無効化

-xtrigraphs オプションは、ISO/ANSI C 標準規格で定義されているようにコンパイラが 3 文字表記シーケンスを認識するかどうかを決定します。

デフォルトでは、コンパイラは -xtrigraphs=yes と仮定し、コンパイル単位全体に渡ってすべての 3 文字表記シーケンスを認識します。

コンパイラが 3 文字表記シーケンスとして解釈している疑問符 (?) を含むリテラル文字列がソースコードに存在する場合、-xtrigraph=no サブオプションを使用して 3 文字表記シーケンスの認識を無効にできます。-xtrigraphs=no オプションは、コンパイル単位全体に渡ってすべての 3 文字表記の認識を無効にします。

次の trigraphs_demo.c というソースファイルの例を考えてみましょう。

```
#include <stdio.h>

int main ()
{

  (void) printf("(\?\?) in a string appears as (??)\n");

  return 0;
}
```

-xtrigraphs=yes を指定してこのコードをコンパイルすると、次のように出力されます。

```
example% cc -xtrigraphs=yes trigraphs_demo.c
example% a.out
(??) in a string appears as (]
```

-xtrigraphs=no を指定してコンパイルした場合は、次のように出力されます。

```
example% cc -xtrigraphs=no trigraphs_demo.c
example% a.out
(??) in a string appears as (??)
```

-xtrigraphs オプションの使用方法の詳細は、cc(1) のマニュアルページを参照してください。3 文字表記については、『C ユーザーズガイド』の ANSI/ISO C への移行に関する章を参照してください。

先読み潜在期間の指定子

(SPARC プラットフォーム) 大規模マルチプロセッサ上で計算量の多いコードを実行している場合は、`-xprefetch` の新しいサブオプション `latx:`*factor* を使用すると便利な場合があります。このサブオプションは、指定された係数に従って先読みから関連するロードまたはストアに至るまでのデフォルトの待ち時間を調整するようにコードジェネレータに指示します。

詳細は、cc(1) のマニュアルページを参照してください。

`-I-` オプションによるデフォルト検索パスの無効化

新しい `-I-` オプションを使用すると、コンパイラがインクルードファイルの検索に使用するアルゴリズムを細かく制御できます。この節では、まずデフォルトの検索アルゴリズムについて説明し、続いてこれらのアルゴリズムに対する `-I-` の効果について説明します。

引用符で囲まれたファイルのデフォルトの検索アルゴリズム

`#include "foo.h"` という形式の (二重引用符が使用される) インクルードファイルの場合は、コンパイラは次の順に検索します。

1. 現在のディレクトリ (つまり「インクルードしている」ファイルが存在するディレクトリ)
2. `-I` オプションで指定されているディレクトリ (存在する場合)
3. `/usr/include` ディレクトリ

山括弧で囲まれたファイルのデフォルトの検索アルゴリズム

`#include <foo.h>` という形式の (山括弧が使用される) インクルードファイルの場合は、コンパイラは次の順に検索します。

1. `-I` オプションで指定されているディレクトリ (存在する場合)
2. `/usr/include` ディレクトリ

-I- オプションによる検索アルゴリズムの変更

新しい -I- オプションを使用すると、デフォルトの検索規則を細かく制御できます。コマンド行に -I- が指定されると、コンパイラは次のように動作します。

- -I でディレクトリが明示的に指定されない限り、現在のディレクトリを検索しません。この規則は、#include "foo.h" のようなインクルードファイルについても同様に適用されます。
- #include "foo.h" という形式のインクルードファイルの場合は、次の順に検索します。

  a. -I オプションで指定されているディレクトリ (-I- の前と後の両方)

  b. /usr/include ディレクトリ

- #include <foo.h> という形式のインクルード文の場合、インクルードファイルを次の順に検索します。

  a. -I- の後の -I オプションで指定されるディレクトリ (つまり、-I- の前に現れる -I ディレクトリは検索しない)

  b. /usr/include ディレクトリ

次の例は、prog.c をコンパイルする場合に -I- を使用した結果を示しています。

prog.c	`#include "a.h"`
	`#include <b.h>`
	`#include "c.h"`

c.h	`#ifndef _C_H_1`
	`#define _C_H_1`
	`int c1;`
	`#endif`

int/a.h	#ifndef _A_H #define _A_H #include "c.h" int a; #endif
int/b.h	#ifndef _B_H #define _B_H #include <c.h> int b; #endif
int/c.h	#ifndef _C_H_2 #define _C_H_2 int c2; #endif

次のコマンドは、現在のディレクトリ (インクルードしているファイルのディレクトリ) から #include "foo.h" という形式のインクルードファイルを検索するデフォルトの動作を示しています。inc/a.h 内の #include "c.h" 文を処理する場合、プリプロセッサは inc サブディレクトリから c.h ヘッダーファイルをインクルードします。prog.c 内の #include "c.h" 文を処理する場合は、プリプロセッサは prog.c を含んでいるディレクトリから c.h ファイルをインクルードします。-H オプションは、インクルードされるファイルのパスを出力するようにコンパイラに指示します。

```
example% cc -c -Iinc -H prog.c
inc/a.h
        inc/c.h
inc/b.h
        inc/c.h
c.h
```

次のコマンドは、-I- オプションの効果を示しています。プリプロセッサは、#include "foo.h" という形式の文を処理する場合にインクルードしているディレクトリを第一候補として検索しません。代わりに、-I オプションで指定されるディレ

クトリをコマンド行に現れる順に検索します。inc/a.h 内の #include "c.h" 文を処理する場合は、inc/c.h ヘッダーファイルではなく ./c.h ヘッダーファイルをインクルードします。

```
example% cc -c -I. -I- -Iinc -H prog.c
inc/a.h
        ./c.h
inc/b.h
        inc/c.h
./c.h
```

詳細は、cc(1) のマニュアルページの -I- の説明を参照してください。

C++ コンパイラ

表 2-2 は、C++ コンパイラ (5.2) の Sun WorkShop 6 update 1 リリースで使用できる新機能を示しています。これらの機能の一部は、表の次の節でさらに詳しく説明しています。

表 2-2　C++ コンパイラの新機能

機能	オプション	説明
UltraSPARC III プロセッサのサポート	`-xtarget` `-xchip`	新リリースの `-xtarget` および `-xchip` オプションには、`ultra3` を指定できるようになりました。UltraSPARC III で最良のパフォーマンスを得るために推奨されるフラグについては、次の節の説明を参照してください。
コンパイル時のパフォーマンス		サイズの大きいプログラムの場合、コンパイル速度がかなり高速になります。とりわけ、テンプレートを多用するプログラムでこの効果が顕著に現れます。
一時オブジェクトの寿命	`-features=tmplife`	(標準モードのみ) 新しい `-features=tmplife` サブオプションは、C++ 標準規格の条件に従って一時オブジェクトを破棄するようにコンパイラに指示します。
`-I-` オプションによるデフォルト検索パスの無効化	`-I-`	新しい `-I-` オプションを使用すると、コンパイラがインクルードファイルの検索時に使用するアルゴリズムを細かく制御できます。
C++ に対する区間演算サポート	`-xia` `-library=[no%]interval`	(SPARC プラットフォーム) C++ コンパイラのこのリリースは、区間演算ライブラリに対する C++ インタフェースを提供しています。

表 2-2　C++ コンパイラの新機能 (続き)

機能	オプション	説明
複数言語のリンク	`-xlang` `-staticlib`	新しい `-xlang` オプションは、Fortran オブジェクトファイルと C++ オブジェクトファイルが混在したリンクを行います。
3 文字表記認識の有効化と無効化	`-xtrigraphs`	新しい `-xtrigraphs` オプションでは、3 文字表記変換を行うかどうか指定できます。
リンカーエラーメッセージのフィルタ処理	`-filt`	新しい `-filt` オプションでは、リンカーエラーメッセージのフィルタ処理をカスタマイズできします。たとえば、 符号化された名前を要求できます。
共有 libCstd		`libCstd` の共有バージョンが、C++ コンパイラの `lib` ディレクトリに含まれています。
共有 libiostream		`libiostream` の共有バージョンが、C++ コンパイラの `lib` ディレクトリに含まれています。
最適化プラグマ		C++ コンパイラは、新しい最適化プラグマである `no_side_effects` と `returns_new_memory` を認識します。
拡張子 .c++ の認識		新リリースの C++ コンパイラは、`.c++` を有効なファイル名接尾辞として認識します。
先読み潜在期間の指定子	`-xprefetch=...,latx;`*`fact or`*	この新しいサブオプションによって、データを先読みするためにコンパイラが生成する命令と、その後使用するロードまたはストア命令のデータがどれくらい時間的に離れているか指定できます。

UltraSPARC III プロセッサのサポート

UltraSPARC III プロセッサ上でプログラムをコンパイルして実行する場合に最良のパフォーマンスを得るには、以下のオプションを使用してコンパイルしてください。

```
-fast -xcrossfile -xprofile={collect:|use:}
```

- クロスコンパイルを行う (UltraSPARC III 以外のプラットフォームでコンパイルするが、UltraSPARC III システムで動作するようにオブジェクトバイナリを生成する) には、正しいキャッシュサイズと最適化方針が設定されるように以下のオプションを追加してください。

  ```
  -xtarget=ultra3 -xarch={v8plusb|v9b}
  ```

- 32 ビットコードの生成には `-xarch=v8plusb`、64 ビットコードの生成には `v9b` を指定してください。`-xarch=v9b` は 64 ビットコードを生成しなければならない場合だけに使用することをお勧めします。状況によっては、`-xarch=v9b` を使用したためにパフォーマンスが低下することがあります。

- `-fast` オプションは、実行速度を重視する最適化を有効にし、最適化レベルを `-xO5` に引き上げます。`-fast` を使用すると、可能な場合により高速な算術演算を代用できるようにするため、`-fsimple=2` を使用します。しかしこの場合、多少の誤りが発生する可能性があることに注意してください。

- `-xcrossfile` が指定されると、コンパイラはコマンド行に指定されたすべてのソースファイルに最適化 (中間手続き的な最適化を含む) を適用します。

- `-xprofile={collect:|use:}` は、プログラムパフォーマンスのプロファイルを有効にします。プロファイルによりコンパイラは、コードの中で最も頻繁に実行されているセクションを特定し、最良の効果を得るための局所的な最適化を行います。

注 - `-xarch={v8plusb|v9b}` により UltraSPARC III プラットフォーム向けにコンパイルされたプログラムは、UltraSPARC III 以外のプラットフォームでは動作しません。UltraSPARC I、UltraSPARC II、および UltraSPARC III の各プラットフォームで互換性を維持したまま動作するプログラムをコンパイルするには、`-xarch={v8plusa|v9a}` を使用してください。

一時オブジェクトの寿命

式を評価する場合に、コンパイラは一時オブジェクトを作成することがあります。たとえば、関数が呼び出された場合や、キャストがクラスオブジェクトを生成する場合がそうです。古い言語定義では、一時オブジェクトが生成されたブロックの最後に到達するまでの任意の時点でこのような一時オブジェクトを破棄できました。しかし、C++ 標準規格では、一時オブジェクトが参照の初期化に使用されないかぎり、そのオブジェクトが生成されている式全体の最後で、そのオブジェクトを破棄しなければならないと定めています。

デフォルトでは、C++ コンパイラ (5.2) は、古い言語定義を実装します。つまり、一時オブジェクトが作成されたブロックの最後でそのオブジェクトを破棄します。コンパイラに C++ 標準規格の条件に従って一時オブジェクトを破棄させるには、新しい `-features=tmplife` サブオプションを使用してください。

例として、次のブロックを考えてみましょう。

```
{  foo(ClassA()); bar(ClassB()); some-statement; }
```

コマンド行で `-features=tmplife` を指定した場合、呼び出し順序は次のようになります。

```
tmp1=ClassA(); foo(tmp1); tmp1.~ClassA();
tmp2=ClassB(); bar(tmp2); tmp2.~ClassB();
some-statement;
```

`-features=tmplife` サブオプションを指定しない場合、呼び出し順序は次のようになります。

```
tmp1=ClassA(); foo(tmp1);
tmp2=ClassB(); bar(tmp2);
some-statement;
tmp2.~ClassB();
tmp1.~ClassA();
```

このサブオプションは、互換モード (`-compat[=4]`) では使用できません。

-I- オプションによるデフォルト検索パスの無効化

新しい -I- オプションを使用すると、コンパイラがインクルードファイルの検索時に使用するアルゴリズムを細かく制御できます。この節では、まずデフォルトの検索アルゴリズムについて説明し、続いてこれらのアルゴリズムに対する -I- の効果について説明します。

引用符で囲まれたファイルのデフォルトの検索アルゴリズム

#include "foo.h" という形式の (二重引用符が使用される) インクルードファイルの場合は、コンパイラは次の順に検索します。

1. 現在のディレクトリ (つまり「インクルードしている」ファイルが存在するディレクトリ)
2. -I オプションで指定されているディレクトリ (存在する場合)
3. コンパイラによって提供される C++ ヘッダーファイル、ANSI C ヘッダーファイル、および特殊目的ファイルの各ディレクトリ
4. /usr/include ディレクトリ

山括弧で囲まれたファイルのデフォルトの検索アルゴリズム

#include <foo.h> という形式の (山括弧が使用される) インクルードファイルの場合派、コンパイラは次の順に検索します。

1. -I オプションで指定されているディレクトリ (存在する場合)
2. コンパイラによって提供される C++ ヘッダーファイル、ANSI C ヘッダーファイル、および特殊目的ファイルの各ディレクトリ
3. /usr/include ディレクトリ

注 - インクルードファイルの名前が標準ヘッダーの名前と一致する場合は、『C++ ユーザーズガイド』の第 5 章の「標準ヘッダーの実装」の節も参照してください。

-I- オプションによる検索アルゴリズムの変更

新しい -I- オプションを使用すると、デフォルトの検索規則を細かく制御できます。コマンド行に -I- が指定されると、コンパイラは次のように動作します。

- -I 指令でディレクトリが明示的に指定されない限り、現在のディレクトリを検索しません。この規則は、#include "foo.h" という形式のインクルードファイルについても同様に適用されます。
- #include "foo.h" という形式のインクルードファイルの場合は、次の順に検索します。

  a. -I オプション (-I- の前と後の両方) で指定されたディレクトリ

  b. コンパイラによって提供される C++ ヘッダーファイル、ANSI C ヘッダーファイル、および特殊目的ファイルの各ディレクトリ

  c. /usr/include ディレクトリ

- #include <foo.h> という形式のインクルードファイルの場合は、次の順に検索します。

  a. -I- の後の -I オプションで指定されるディレクトリ (つまり、-I- の前に現れる -I ディレクトリは検索しない)

  b. コンパイラによって提供される C++ ヘッダーファイル、ANSI C ヘッダーファイル、および特殊目的ファイルの各ディレクトリ

  c. /usr/include ディレクトリ

次の例は、prog.cc でコンパイルする場合に -I- を使用した結果を示しています。

prog.cc	#include "a.h" #include <b.h> #include "c.h"
c.h	#ifndef _C_H_1 #define _C_H_1 int c1; #endif
inc/a.h	#ifndef _A_H #define _A_H #include "c.h" int a; #endif
inc/b.h	#ifndef _B_H #define _B_H #include <c.h> int b; #endif
inc/c.h	#ifndef _C_H_2 #define _C_H_2 int c2; #endif

次のコマンドは、現在のディレクトリ (インクルードしているファイルのディレクトリ) から #include "foo.h" という形式のインクルードファイルを検索するデフォルトの動作を示しています。inc/a.h 内の #include "c.h" 文を処理する場合、コンパイラは inc サブディレクトリから c.h ヘッダーファイルをインクルードします。prog.c 内の #include "c.h" 文を処理する場合は、プロプロセッサは、

prog.c を含んでいるディレクトリから c.h ファイルをインクルードします。-H オプションは、インクルードされるファイルのパスを出力するようにコンパイラに指示します。

```
example% CC -c -Iinc -H prog.cc
inc/a.h
        inc/c.h
inc/b.h
        inc/c.h
c.h
```

次のコマンドは、-I- オプションの効果を示しています。コンパイラは、#include "foo.h" という形式の文を処理する場合にインクルードしているディレクトリを第一候補として検索しません。代わりに、-I オプションで指定されるディレクトリをコマンド行に現れる順に検索します。inc/a.h 内の #include "c.h" 文を処理する場合は、inc/c.h ヘッダーファイルではなく ./c.h ヘッダーファイルをインクルードします。

```
example% CC -c -I. -I- -Iinc -H prog.cc
inc/a.h
        ./c.h
inc/b.h
        inc/c.h
./c.h
```

詳細は、CC(1) のマニュアルページに含まれる -I- の説明を参照してください。

C++ に対する区間演算サポート

(SPARC プラットフォーム) Sun WorkShop™ 6 update 1 C++ Compilers (5.2) は、C++ 区間演算ライブラリに対する C++ インタフェースを提供します。詳細は、このマニュアルの 49 ページの「区間演算」と、http://docs.sun.com (Forte™ Developer 6 update 1/Sun WorkShop 6 update 1 Collection) に掲載されている『C++ Interval Arithmetic Programming Reference』を参照してください。

関連するコンパイラオプションの詳細は、CC(1) のマニュアルページに含まれる次のオプションの説明を参照してください。

- -xia
- -library=[no%]interval

注 - C++ 区間演算ライブラリは、Fortran コンパイラで実装されている区間演算と互換性があります。

複数言語のリンク

新しい `-xlang` オプションを使用すると、Fortran オブジェクトファイルと C++ オブジェクトファイルをリンクできます。たとえば、C++ の主プログラムと Fortran オブジェクトファイルをリンクできます。

複数の言語を同時にリンクする場合に使用するドライバを決定するには、次の言語階層を使用してください。

1. C++
2. Fortran 95 (または Fortran 90)
3. Fortran 77

Fortran 95、Fortran 77、および C++ のオブジェクトファイルを一緒にリンクするには、最上位の言語のドライバを使用してください。たとえば、C++ と Fortran 95 の各オブジェクトファイルをリンクするには、次の C++ コンパイラコマンドを使用します。

```
example% CC -xlang=f95 ...
```

Fortran 95 と Fortran 77 の各オブジェクトファイルをリンクするには、次のように Fortran 95 ドライバを使用します。

```
example% f95 -xlang=f77 ...
```

詳細は、CC(1) のマニュアルページに含まれる以下のオプションの説明を参照してください。

- `-xlang=f77、f90、f95`
- `-staticlib=[no%]f77、[no%]f90、[no%]f95、[no%]sunperf`

3 文字表記認識の有効化と無効化

-xtrigraphs オプションは、ISO/ANSI C 標準規格で定義されているようにコンパイラが 3 文字表記シーケンスを認識するかどうかを決定します。

デフォルトでは、コンパイラは -xtrigraphs=yes と仮定し、コンパイル単位全体に渡ってすべての 3 文字表記シーケンスを認識します。

コンパイラが 3 文字表記シーケンスとして解釈している疑問符 (?) を含むリテラル文字列がソースコードに存在する場合、-xtrigraph=no サブオプションを使用して 3 文字表記シーケンスの認識を無効にできます。-xtrigraphs=no オプションは、コンパイル単位全体に渡ってすべての 3 文字表記の認識を無効にします。

次の trigraphs_demo.cc というソースファイルの例を考えてみましょう。

```
#include <stdio.h>

int main ()
{

  (void) printf("(\?\?) in a string appears as (??)\n");

  return 0;
}
```

-xtrigraphs=yes を指定してこのコードをコンパイルすると、次のように出力されます。

```
example% CC -xtrigraphs=yes trigraphs_demo.cc
example% a.out
(??) in a string appears as (]
```

-xtrigraphs=no を指定してコンパイルした場合は、次のように出力されます。

```
example% CC -xtrigraphs=no trigraphs_demo.cc
example% a.out
(??) in a string appears as (??)
```

-xtrigraphs オプションの使用の詳細は、CC(1) のマニュアルページを参照してください。3 文字表記については、『C ユーザーズガイド』の ANSI/ISO C への移行に関する章を参照してください。

リンカーエラーメッセージのフィルタ処理

新しい -filt オプションは、通常、CC がリンカーからのエラーメッセージに対して行うフィルタ処理を抑止します。

- -filt=no%names は、C++ の符号化された C++ リンカー名を復号化しません。
- -filt=no%returns は、関数の戻り型を復号化を抑止しません。これにより関数名をより簡単に特定できます。ただし、仮想関数の場合、関数名は同じで戻り型のみが異なる関数があるため、注意が必要です。
- -filt=no%errors は、リンカーからのエラーメッセージに関する C++ の説明を表示しません。説明を抑止すると、リンカーの診断が別のツールに直接渡される場合に役立ちます。

次の例は、次のコードをコンパイルする場合に -filt を使用する効果を示しています。

```
// filt_demo.cc
class type {
public:
    virtual ~type(); // 何も定義されていません
};

int main()
{
    type t;
}
```

このコードを -filt オプションなしでコンパイルすると、コンパイラは -filt=names,returns,errors と仮定し、以下のような標準の出力を表示します。

```
example% CC filt_demo.cc
未定義の            最初に参照している
シンボル                 ファイル
type::~type()          filt_demo.o
type::__vtbl           filt_demo.o
[ヒント: クラス型の、インライン化されておらず、純粋でない 1 つ目の仮想関数が
定義されているか確認してください。]

ld: 重大なエラー: シンボル参照エラー。a.out に書き込まれる出力はありません。
```

注意 - 上記のヒントは、日本語環境では技術的制限のため出力されません。

次のコマンドは、符号化された C++ リンカー名の復号化を抑止するとともに、リンカーエラーの C++ 説明を抑止します。

```
example% CC -filt=no%names,no%errors filt_demo.cc
未定義の                          最初に参照している
シンボル                               ファイル
__1cEtype2T6M_v_                     filt_demo.o
__1cEtypeG__vtbl_                    filt_demo.o
ld: 重大なエラー: シンボル参照エラー。a.out に書き込まれる出力はありません。
```

詳細は、CC(1) のマニュアルページを参照してください。

共有 libCstd

Sun WorkShop 6 update 1 Compilers C++ (5.2) には、libCstd ライブラリの共有バージョンが含まれます。

libCstd の共有バージョンを使用するには、コンパイルとリンクを別々に行ってください。そして、リンク時には、次の例に示すように -library=no%Cstd オプションを使用してコマンド行に共有ライブラリ名を明示的に指定してください。

```
example% CC -library=no%Cstd *.o \
-o myprog /opt/SUNWspro/WS6U1/lib/libCstd.so.1
```

Sun WorkShop 6 update 1 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。

注 – C++ ソースコードをコンパイルするコマンド内に -library=no%Cstd を使用すると、コンパイラは C++ の標準ヘッダーを検索できません。

共有 libiostream

Sun WorkShop 6 update 1 C++ Compilers (5.2) には、従来方式の iostream ライブラリの共有バージョン、libiostream が含まれています。

この共有バージョン libiostream を使用するには、コンパイルとリンクを別々に行ってください。そしてリンク時には、次の例に示すようにコマンド行に共有ライブラリ名を明示的に指定し、-library=iostream オプションは使用しないでください。

```
example% CC *.o -o myprog \
/opt/SUNWspro/WS6U1/lib/lib/libiostream.so.1
```

Sun WorkShop 6 update 1 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。

注 - プログラム構築における各コンパイルでは -library=iostream を使用する必要がありますが、リンク時には -library=iostream を使用してはなりません。

最適化プラグマ

オプティマイザがより優れたコードを生成するように、以下の新しいプラグマを使用できます。

- #pragma no_side_effect (*name* ...)

  このプラグマは、関数が持続的な状態を変更しないことを示すために使用してください。

- #pragma returns_new_memory (*name*, ...)

このプラグマは、関数が新しく割り当てられたメモリーのアドレスを返し、かつポインタがほかのポインタを使用して別名を設定しないことが確実である場合に使用してください。

どちらのプラグマも、*name* はその名前を使用する、直前に宣言されている関数を示します。

これらのプラグマは、それらが参照する関数の直後に置いてください。たとえば、次の最初のコード例は `no_side_effects` プラグマの正しい配置を示しており、2 つ目の例は誤った配置を示しています。

```
class good_example {
  void no_op();
  #pragma no_side_effects (no_op) // 正しい配置
}

class bad_example{
  void no_op();
}
#pragma no_side_effects (no_op) // 誤った配置
```

拡張子 `.c++` の認識

コマンド行上にファイル名が出現する場合、コンパイラは接尾辞を見てそのファイルをどのように処理するかを決定します。たとえば、`.o` で終わっているファイルの場合、コンパイラはオブジェクトファイルとして処理します。Sun WorkShop 6 update 1 C++ Compilers (5.2) は、以下の拡張子を持つファイルを C++ のソースファイルとして認識します。

- `.c`
- `.C`
- `.cc`
- `.cpp`
- `.cxx`
- `.c++`
- `.i`

先読み潜在期間の指定子

(SPARC プラットフォーム) 大規模マルチプロセッサ上で計算量の多いコードを実行している場合は、`-xprefetch` の新しいサブオプション `latx:`*factor* を使用すると便利な場合があります。このサブオプションは、指定された係数に従って先読みから関連するロードまたはストアに至るまでのデフォルトの待ち時間を調整するようにコードジェネレータに指示します。

詳細は、`cc`(1) のマニュアルページを参照してください。

Fortran コンパイラ

Sun WorkShop 6 update 1 では、Fortran 95 コンパイラと Fortran 77 コンパイラの両方がリリースされています。

表 2-3 は、Sun WorkShop 6 update 1 リリースで使用できる新機能のうち Fortran 95 コンパイラと Fortran 77 コンパイラの両方に共通する機能を示しています。これらの機能の一部は、表の次の節でさらに詳しく説明しています。

表 2-3　Fortran コンパイラの新機能

機能	オプション	説明
UltraSPARC III プロセッサのサポート	`-xtarget` `-xchip`	新リリースの `-xtarget` および `-xchip` オプションには、`ultra3` を指定できるようになりました。UltraSPARC III で最良のパフォーマンスを得るために推奨される使用フラグについては、次の節の説明を参照してください。
`int2` 組み込みのサポート		`int2` 組み込みは、古い Fortran 77 プログラムとの互換性をサポートします。これは、データ引数 *var* を 2 バイト整数に変換する上で推奨される Fortran 95 の `int` (*var*,2) 組み込みと同じ機能を持ちます。
`-fast` オプションの拡張	`-fast`	新リリースの `-fast` オプションには、`-xprefetch` オプションが含まれます。
先読み潜在期間の指定子	`-xprefetch=...,latx:`*factor*	この新しいサブオプションによって、データを先読みするためにコンパイラが生成する命令と、その後使用するロードまたはストア命令のデータがどれくらい離れているかを時間的に指定できます。
複数言語のリンク	`-xlang`	新しい `-xlang` オプションは、Fortran のオブジェクトファイルと C++ のオブジェクトファイルが混在したリンクを行います。

UltraSPARC III プロセッサのサポート

UltraSPARC III プロセッサ上でプログラムをコンパイルして実行する場合に最良のパフォーマンスを得るには、以下のオプションを使用してコンパイルしてください。

```
-fast -xcrossfile -xprofile={collect:|use:}
```

- クロスコンパイルを行う (UltraSPARC III 以外のプラットフォームでコンパイルするが、UltraSPARC III システムで動作するようにオブジェクトバイナリを生成する)には、正しいキャッシュサイズと最適化方針が設定されるように以下のオプションを追加してください。

  ```
  -xtarget=ultra3 -xarch={v8plusb|v9b}
  ```

- 32 ビットコードの生成には `-xarch=v8plusb`、64 ビットコードの生成には `v9b` を指定してください。非常に大きなデータファイルにアクセスするプログラムの場合、64 ビットコードの方が優れたパフォーマンスを得られます。しかし、`-xarch=v9b` は 64 ビットコードを生成しなければならない場合だけに使用することをお勧めします。状況によっては、`-xarch=v9b` を使用したためにパフォーマンスが低下することがあります。

- `-fast` オプションは、実行速度を重視する最適化を有効にし、最適化レベルを `-xO5` に引き上げます。`-fast` を使用すると、可能な場合により高速な算術演算を代用できるようになるため、`-fsimple=2` を使用します。しかしこの場合、多少の誤りが発生する可能性があることに注意してください。

- `-xcrossfile` が指定されると、コンパイラはコマンド行に指定されたすべてのソースファイルに最適化 (中間手続き的な最適化を含む) を適用します。

- `-xprofile={collect:|use:}` は、プログラムパフォーマンスのプロファイルを有効にします。プロファイルによりコンパイラは、コードの中で最も頻繁に実行されているセクションを特定し、最良の効果を得るための局所的な最適化を行います。

注 - `-xarch={v8plusb|v9b}` を指定して UltraSPARC III プラットフォーム向けにコンパイルされたプログラムは、UltraSPARC III 以外のプラットフォームでは動作しません。UltraSPARC I、UltraSPARC II、および UltraSPARC III の各プラットフォームで互換性を維持して動作するプログラムをコンパイルするには、`-xarch={v8plusa|v9a}` を使用してください。

---

int2 組み込みのサポート

新リリースの Fortran 95 コンパイラと Fortran 77 コンパイラは、2 バイト整数へのデータ型変換に int2 組み込みをサポートします。int2 の組み込みとしての使用は、多くの古い Fortran 77 コードでは M=int2(J) という書式で出現しています。

Fortran 95 では、M=int(J,2) を使用することをお勧めします。

-fast オプションの拡張

-fast オプションに含められるオプションの一覧に -xprefetch オプションが追加されました。これにより、コンパイラは先読み指令を戦略的に生成できるようになりました。-xprefetch を使用すると、データ処理ループを持つコードでパフォーマンスをかなり高めることができます。UltraSPARC III プラットフォームの先読み機構は、UltraSPARC II プラットフォームで使用された機構よりもはるかに改良されています。

先読み潜在期間の指定子

(SPARC プラットフォーム) 大規模マルチプロセッサ上で計算量の多いコードを実行している場合は、-xprefetch の新しいサブオプション latx:*factor* を使用すると便利な場合があります。このサブオプションは、指定された係数に従って先読みから関連するロードまたはストアに至るまでのデフォルトの待ち時間を調整するようにコードジェネレータに指示します。

詳細は、f77(1) と f95(1) のマニュアルページを参照してください。

複数言語のリンク

新しい -xlang オプションを使用すると、Fortran オブジェクトファイルと C++ オブジェクトファイルをリンクできます。たとえば、Fortran の主プログラムを C++ オブジェクトファイルにリンクできます。

複数の言語を同時にリンクする場合に使用するドライバを決定するには、次の言語階層を使用してください。

1. C++
2. Fortran 95

3. Fortran 77

Fortran 95、Fortran 77、および C++ のオブジェクトファイルを一緒にリンクするには、最上位の言語のドライバを使用してください。たとえば、C++ と Fortran 95 の各オブジェクトファイルをリンクするには、次の C++ コンパイラコマンドを使用します。

```
example% CC -xlang=f95 ...
```

Fortran 95 と Fortran 77 の各オブジェクトファイルをリンクするには、次のように Fortran 95 ドライバを使用します。

```
example% f95 -xlang=f77 ...
```

区間演算

表 2-4 は、Sun WorkShop 6 update 1 リリースの区間演算で使用できる新機能を示しています。

表 2-4　区間演算の新機能

機能	説明
C++ に対する区間演算サポート	(SPARC プラットフォーム) このリリースの C++ コンパイラは、区間演算ライブラリに対する C++ インタフェースを提供します。
新しい f95 INTERVAL 組み込み演算子と関数	`f95` 区間演算では、依存型の減算演算子、積集合関数による除算、および汎用的な乱数生成サブルーチン `RANDOM_NUMBER` の区間バージョンに対するサポートが追加されています。

C++ に対する区間演算サポート

Sun WorkShop 6 update 1 リリースには、Fortran 95 区間演算に含まれる区間関数と区間演算子の C++ バージョンが含まれています。C++ 区間演算ライブラリの詳細は、`http://docs.sun.com` (Forte™ Developer 6 update 1/Sun WorkShop 6 update 1 Collection) に掲載されている『C++ Interval Arithmetic Programming Reference』を参照してください。

C++ の区間演算サポートは、C++ ヘッダーファイルと 3 つの区間クラス (float、double、および long double にそれぞれ 1 つ) を提供するライブラリを提供します。区間クラスには次のものが含まれます。

- 閉じた数学的システムを形成する区間算術演算と数学関数。これは、演算子とオペランドのあらゆる組み合わせ (ゼロおよび無限が関係するゼロ除算などの不確定な形式を含む) に対して有効な結果が生成されることを意味します。
- 次に示す 3 種類の区間関係を示す関数:
  - 断定的な関係
  - 可能性のある関係
  - 集合の関係
- `intersect` や `interval_hull` などの区間固有の関数
- `inf`、`sup`、`wid` などの区間固有の関数
- 区間入出力 (単一数値の入出力など)

標準モードのコンパイルの場合、ライブラリ内のシンボルはすべて名前空間 `SUNW_interval` にあります。

コンパイルインタフェースは、次の要素から構成されます。

- `-library` フラグ用の新しい値 `interval`

  適切なライブラリに展開されます
- `-staticlib` フラグ用の新しい値 `interval`

  このリリースでは静的ライブラリだけが提供されているため、現在は無視されます
- 新しいフラグ `-xia`

  Fortran コンパイラで使用されるフラグと同じ (展開は異なります)

C++ 区間演算機能を使用するには、コードに次のヘッダーファイルを追加してください。

```
#include <suninterval.h>
```

次に、-xia コマンド行オプションを使用してコードをコンパイルする例を示します。

```
example% CC -o filename -xia filename.cc
```

新しい f95 INTERVAL 組み込み演算子と関数

このリリースでは、以下の機能が f95 区間演算に追加されました。

- 依存型の減算演算子
- 積集合関数による除算
- 乱数サブルーチン

依存型の減算演算子

依存型の減算演算子 .DSUB. は、区間演算加算の一方のオペランドの回復に使用できます。

一方の区間変数が他方の区間変数に適用された算術演算の結果である場合、これらの 2 つの区間変数は依存関係にあります。たとえば、X = A + B の場合、X は A と B の両方によって決まります。X から A または B を回復させる場合、依存型の区間減算はより狭幅の区間結果を生成します。

注 - 定数は独立しているため、依存型の演算を区間定数に適用することはできません。依存型の演算を区間定数に適用すると、コンパイル時エラーが発生します。

X.DSUB.A の結果には、区間等式 X = A + B の B に対する解が含まれます。解が存在しない場合、結果は [-inf,inf] です。

- 引数: X と A は同じ種類の型パラメータ値を持つ区間でなければならない
- 結果の型: X と同じ

以下の例は、DSUB の動作を示しています。

- $-\infty < \underline{x} \le \bar{x} < +\infty$ の場合、X .DSUB. X = [0]
- WID(X) < WID(A) の場合、X .DSUB. A = [-inf,inf]。これは、X = A + B の場合、WID(X) が WID(A) 以上でなければならないため
- [empty] .DSUB. [empty] = [-inf,inf]
- $A \neq$ [empty] の場合、[empty] .DSUB. A = [empty]

- X ≠ [empty] の場合、X .DSUB. [empty] = [-inf,inf]。この結果になるのは、X ≠ [empty] と A ≠ [empty] が不可能な組み合わせのため

積集合関数による除算

関数 DIVIX は、区間 C と区間除算演算 (B/A) の共通部分の区間格納装置である C ∩ (B/A) を返します。

A にゼロが含まれる場合には、区間除算演算 (B/A) の数学上の結果は互いに素である 2 つの区間の和集合です。和集合内の各区間は、現在実装されている区間演算システムで表現できます。DIVIX 関数は、これらの区間の 1 つまたは一部を返すことができます。

- **引数**: A、B、および C は、同じ種類の型パラメータ値を持つ区間でなければならない
- **戻り値**: A と同じ

次の例は、DIVIX の出力を示しています。

```
DIVIX( [3.0,5.0] , [6.0,15.0] , [2.0,6.0] )== [2.0,5.0]
DIVIX( [-3.0,5.0] , [8.0,20.0] , [-0.5,1.0] )== [EMPTY]
DIVIX( [-3.0,5.0] , [8.0,20.0] , [-7.0,8.0] )== [-7.0,8.0]
DIVIX ([-1,1], [1], [-Inf,0.0E+0]) == [-Inf,-1.0]
DIVIX ([-1,1], [1], [0,inf])       == [1.0,Inf]
```

乱数サブルーチン

RANDOM_NUMBER(HARVEST) は、区間変数 HARVEST を通して 1 つまたは複数の疑似乱数区間を返します。$[\underline{x}, \bar{x}]$

$0 \le \underline{x} \le \bar{x} < 1$ は、区間終了位置 ($\underline{x}$ は下位の終了位置、$\bar{x}$ は上位の終了位置) について当てはまります。そのため $\underline{x}$ は区間 [0, 1] に均等に分配され、$\underline{x}$, $\bar{x}$ が与えられると $\bar{x}$ は区間 $[\underline{x}, 1]$ に均等に分配されます。

dbx

表 2-5 は、dbx の Sun WorkShop 6 update 1 リリースで使用できる新機能を示しています。

表 2-5　dbx の新機能

機能	説明
Fortran における区間演算式のサポート	新リリースでは、Fortran の区間型と区間式がサポートされるようになりました。単純演算 (加算、減算、乗算、除算、否定)、一致、および不一致の各演算が実装されています。

Sun Performance Library

Sun Performance Library™ は、数値を多用する問題 (線形代数など) を解決するための最適化された高速の数学サブルーチンです。Sun Performance Library は、Netlib (http://www.netlib.org) から入手できる一連のパブリックドメインアプリケーションに基づいています。これらのルーチンは、機能が拡張され、 Sun Performance Library としてバンドルされています。

表 2-6 は、Sun Performance Library の Sun WorkShop 6 update 1 リリースで使用できる新機能と拡張された機能を示しています。

表 2-6　Sun Performance Library の新機能

機能	説明
UltraSPARC III プロセッサに対するパフォーマンス強化	シングルプロセッサ型およびマルチプロセッサ型の UltraSPARC III プラットフォーム双方に対する最大限のパフォーマンス
高速フーリエ変換 (FFT) の改良	マニュアルページの変更、コードに対するエラーチェックの強化、およびパフォーマンスの向上が行われています。
FFT ドキュメント	`FFTPACK` および `VFFTPACK` ルーチンについて説明した『Using Sun Performance Library Fast Fourier Transform Routines』が作成されました。『Using Sun Performance Library Fast Fourier Transform Routines』は、`http://docs.sun.com` の Forte™ Developer 6 update 1/Sun WorkShop 6 update 1 Collection で入手できます。
スパースソルバー	パフォーマンスと機能性の向上、拡張されたメッセージングシステム、組み込みの SuperLU
並列化とスケーラビリティの拡張	選択されたルーチンとライブラリに対してデータフロープログラミング手法を使用し、パフォーマンスを向上させます。
Open MP 指令の使用	Open MP 指令に対するサポートが強化されています。
区間行列乗算関数の実装	`f95` 組み込みに対して区間ベースの `MATMUL` が開発されました。`MATMUL` は、`GEMM` および `GEMV` ルーチンの非置換ケースです。

標本アナライザ

表 2-7 は、標本アナライザの Sun WorkShop 6 update 1 リリースで使用できる新機能を示しています。

表 2-7　標本アナライザの新機能

機能	説明
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	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は、軽量プロセス (LWP) が動作している CPU のハードウェアカウンタがオーバーフローする場合に、各 LWP の呼び出しスタックを記録します。
スタンドアロンの collect コマンド	新しい `collect` コマンドは、Sun WorkShop および `dbx` とは関係のないアプリケーション上のパフォーマンスデータを収集します。
MPI アプリケーションのサポート機能の改良	新しい `collect` コマンドでは、メッセージ引き渡しインタフェース (MPI) のサポート機能が拡張されています。
OpenMP (`libmtsk`) アプリケーションのサポート機能の改良	新リリースでは、スレーブスレッドがパラレル領域の最後で同期を待機している時とコードがシリアル領域にあるために待機している時を区別できます。
マップファイル生成機能の改良	新リリースでは、関数リストのソートに使用されているメトリック (計測データ) にかかわらず実行可能ファイルに命令できるようにマップファイルが生成されます。
「フィルタを選択」ダイアログへの追加	「実験ファイル」リスト、および「フィルタを選択」ダイアログ内の「すべてを選択」、「すべてを選択解除」、「反転」各ボタンを使用すると、表示されるデータを変更する実験を選択できます。「すべてを有効」、「選択中のすべてを有効」、「すべてを無効」、および「選択中のすべてを無効」ボタンは、実験のデータ表示のオン / オフ切り替えに使用できます。

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

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は、軽量プロセス (LWP) が動作している CPU のハードウェアカウンタがオーバーフローする場合に、各 LWP の呼び出しスタックを記録します。記録されるデータには、CPU のタイムスタンプと ID、スレッド、LWP などがあります。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が行えるのは、Solaris 8 オペレーティング環境 SPARC Platform Edition を使用している UltraSPARC III システム上と、Solaris 8 オペレーティング環境 Intel Platform Edition を使用している Intel システム (Pentium II か III) 上だけです。詳細は、Sun WorkShop オンラインヘルプの「ハードウェアカウンタのオーバーフロープロファイルデータ」と「収集するデータの選択」を参照してください。

スタンドアロンの `collect` コマンド

新しい `collect` コマンドは、Sun WorkShop および `dbx` とは関係のないアプリケーション上のパフォーマンスデータを収集します。`collect` コマンドは、収集されるデータの型を指定する引数、実験および実験グループに名前を付ける引数、`exec` システムコールからの出口で対象プロセスを停止したままにしてデバッガがこのプロセスに連結できるように要求する引数などを提供します。詳細は、`collect`(1) のマニュアルページを参照してください。

MPI アプリケーションのサポート機能の改良

新しい `collect` コマンドでは、メッセージ引き渡しインタフェース (MPI) サポート機能が拡張されています。

- `collect` コマンドを使用すると実験グループを指定できます。これにより、MPI 実行の全プロセスの実験がグループ化され、一括処理されます。
- 呼び出し内のリアルタイム遅延が指定されたしきい値を超える場合、同期遅延トレースはさまざまなスレッド同期ルーチンへのすべての呼び出しを記録します。

第3章

Sun WorkShop 6 の新機能のご紹介

この章では、Sun WorkShop™ 6 のコンパイラとツールの新しい機能について説明します。これらの新機能については、表にまとめられています。一部の機能については、各表の次の節で詳しく説明しています。

この章の内容は次のとおりです。

主な特長

以下の機能が、このリリースの特長です。

- ANSI/ISO C++ への準拠の強化
- OpenMP 並列化指令が追加された Fortran 95 コンパイラ
- UltraSPARC™ III のサポート

- より使いやすくなったプログラミング環境
- 新しいパフォーマンス解析ツール
- Fortran 95 区間演算
- インストールの改良
- HTML 形式による マニュアル、マニュアルページ、README、オンラインヘルプ

C コンパイラ

表 3-1 は、Sun WorkShop 6 C コンパイラの新機能です。これらの機能により、C コンパイラと `lint` ソースコードチェッカーがより優れたものとなっています。

表 3-1　C コンパイラの新機能

機能	説明
`__func__`	新リリースの C コンパイラは、あらゆる関数定義に指定される静的な一定の文字配列 `__func__` を事前定義します。この配列は、関数名によって初期化され、静的関数のスコープ配列を使用できる場所であればどこにでも使用できます。たとえば、包含する関数の名前を出力する場合に使えます。
可変引数マクロ	C プリプロセッサは、`#define` マクロについて可変数の引数を受け付けます。マクロ定義の識別子リストの一部として省略符号 (...) が含まれている場合、マクロ起動時に使用される引数は、マクロ定義内のパラメータより多くなります。
`SUNW_MP_THR_IDLE`	新リリースではスレッドがタスクを終了した後や「スリープ状態」時にもシステム資源を使い続けるかどうかを制御する、`SUNW_MP_THR_IDLE` という環境変数を使用できます。
大規模配列	新リリースの C コンパイラは、大規模な配列オブジェクトをサポートします。『C ユーザーズガイド』の付録 A を参照してください。

表 3-1　C コンパイラの新機能 (続き)

機能	説明
-errchk=locfmtchk	新リリースの lint は、新しいフラグ -errchk=locfmtchk を受け付けます。このフラグは、lint の 1 回目のパスにおいて printf に似た書式の文字列があるかどうかを検査します。
lint が受け付ける新しい指令 (PRINTFLIKE (*n*))	新リリースの lint は、ポインタによる printf() ファミリへの呼び出しすべてを識別する新しい指令を受け付けます。こういった、ポインタによる呼び出しの引数整合性を lint によってチェックできます。
-errwarn=t	新リリースの C コンパイラと lint ソースコードチェッカーは、新しいオプション -errwarn=t をサポートします。このオプションが指定されている場合、指定した警告のどれかが発行されるとコンパイラはエラーステータスを出力して終了します。
-errchk	新リリースの lint の -errchk には、新しい値 signext が用意されています。この値は、-errchk=longptr64,signext のように longptr64 オプションと組み合わせて使えます。このオプションは、64 ビットの開発環境への移行を行いやすくするため、符号拡張について警告します。
-xchar_byte_order	-xchar_byte_order オプションは、複数文字の文字定数から成る文字を、指定されたバイト順序で配置することによって整定数を生成します。
-xinline	新リリースの -xinline オプションには、%auto *function_name* と no% *function_name* の 2 つの値が追加されました。%auto は -xO4 の最適化レベルで適用され、それによって -xinline にリストされている関数以外の関数も、自動的にインライン化します。%no *function-name* は、関数 *function-name* をインライン化しないことをコンパイラに指示します。

表 3-1　C コンパイラの新機能 (続き)

機能	説明
`-xmemalign`	新リリースの C コンパイラには、`-xmemalign` というオプションが新たに用意されています。このオプションはメモリーアクセスが境界整列していない可能性に備えて生成されるコードを制御し、アクセスが境界整列していないときにプログラムの動作を制御します。
`-xprefetch`	新リリースでは、`sun_prefetch.h` ヘッダーファイルと `-xprefetch` オプションを使うことによって明示的な先読み命令を指定できます。
`-xvector`	`-xvector` オプションは、ベクトルライブラリ関数への呼び出しを自動生成します。

C++ コンパイラ

表 3-2 では、Sun WorkShop 6 C++ コンパイラのリリースで利用できる新機能を一覧表示します。一部の製品の機能については、表の次の節でさらに詳しく説明しています。

表 3-2　C++ コンパイラの新機能

機能	説明
部分的特殊化	テンプレートは部分的に特殊化することができます。つまり、一部のテンプレートパラメータのみを指定したり、または複数のパラメータを特定の型のカテゴリに制限することができます。
明示的関数テンプレート引数	関数の引数からテンプレートの引数を推察できない場合、新リリースでは `f<`*template args*`>(`*function args*`)` の構文を使って明示的に指定できます。

関数テンプレートの型名でないパラメータ	新リリースでは、以下のような、関数テンプレートの型名でないパラメータをサポートします。
	`template<int I> void foo( int a[I] ) { ... }`
	`template<int I> void foo( mytype<I> m ) { ... }`

表 3-2　C++ コンパイラの新機能 (続き)

機能	説明
メンバーテンプレート	新リリースの標準モードでは、クラスとクラステンプレートにメンバーとしてテンプレートを持たせることができます。
定義分離テンプレート編成に対する制約の解除	新リリースのコンパイラでは、`-instances != extern` (つまり、`-instances=explicit`、`-instances=global`、`-instances=semiexplicit`、または `-instances=static`) における “定義分離テンプレート編成” に対する制約が解除されています 。`-instances` の設定値にかかわりなく、定義検索時に個別のソースファイルを含めるのがコンパイラのデフォルトの動作となっています。
先読み命令	ヘッダーファイル `sun_prefetch.h` とともに新しいオプション `-xprefetch` を使用すると、UltraSPARC II など先読みをサポートするアーキテクチャ (`-xarch=v8plus`, `v9plus`, `v9plusa`, `v9`, `v9a`) では先読み命令を指定できます。
外部インライン関数	このバージョンのコンパイラでは、外部インライン関数を使用できます。インライン関数に局所的な静的データが入っている場合は、すべてのコンパイル単位でその静的データのコピーが 1 つだけ使用されます。ただし、異なる翻訳単位で取得したインライン関数のアドレスは等しいとはみなしません。
静的変数破棄の順序付け	規格により、静的記憶期間のあるオブジェクトを破棄する順序がさらに詳しく定義されました。静的オブジェクトは、作成した順序とは逆の順序で破棄する必要があります。以前の言語の定義では、いくつかの局面での規定が放置されていました。
副集合の初期設定	(中括弧初期設定を使用できる型に関し) クラスオブジェクトの中括弧初期設定を使用している場合、C++ 標準では、それ自体が集合クラス であるメンバーは、自分自身の型の値で初期設定できるようになりました。
自分専用の C++ 標準ライブラリの使い方	`-library=no%Cstd` オプションを指定することで、コンパイラ付属のバージョンの代わりに、自分専用の C++ 標準ライブラリのバージョンを使用できます。

表 3-2　C++ コンパイラの新機能 (続き)

機能	説明
キャッシュバージョン指定	C++ コンパイラには、キャッシュバージョンの違いを検出し、適切なエラーメッセージを出力する機能があります。
ビットフィールドサイズに関する制限の削除	ビットフィールドのサイズを 32 以下にするという制限は削除されました。ビットフィールドのサイズは任意になりました。
関数のポインタと `void*` の相互間の変換に関する警告	以前のコンパイラは関数のポインタと `void*` で相互間の変換に関する警告を必ず出しましたが、現在では `+w2` オプションを使用する場合に限り警告が出されます。

新規オプションおよび変更されたオプション	以下のリストでは、新規オプションと変更されたオプションを示します。
	• 新規オプション
	`-xcrossfile`
	`-Bsymbolic`
	`-features=[no%]strictdestrorder`
	`-template=extdef`
	• 変更されたオプション
	`-fast`
	`-library=[no%]Catd,+p`
	`-ptr,-xprefetch`
	詳細は、『C++ ユーザーズガイド』を参照してください（`http://docs.sun.com` から参照できます）。

部分的特殊化

テンプレートは完全に特殊化できます。つまり、特定のテンプレート引数に対する実装が定義されました。以下のコード例を参照してください。

```
template<class T, class U> class A { ... }; // 初期テンプレート
template<> class A<int, double> { ... }; // 特殊化
```

テンプレートは部分的に特殊化することもできます。つまり、一部のテンプレートパラメータのみを指定したり、複数のパラメータを特定の型のカテゴリに制限することができます。その結果行われた部分的特殊化は、それ自体もテンプレートとして使用できます。以下は、上記の例の初期テンプレートを使用した例です。

- 初期テンプレートパラメータが int 型である場合の特殊テンプレートの定義。以下のコード例を参照してください。

```
template<classU> class A<int> { ... };
```

- 初期テンプレートパラメータが任意のポインタ型である場合の特殊なテンプレートの定義。以下のコード例を参照してください。

```
template<class T, class U> class A<T*> { ... };
```

- 初期テンプレートパラメータが任意の型のポインタのポインタで、2 番目のテンプレートパラメータが char 型である場合の特殊なテンプレートの定義。以下のコード例を参照してください。

```
template<class T> class A<T**, char> { ... };
```

明示的関数テンプレート引数

関数の引数からテンプレートの引数を推察できない場合、f<*template args*>(*function args*) の構文を使って明示的に指定できます。
以下は、その一例です。

```
template<class Mytype> Mytype* construct(float,float);
...
int* x = construct<int>(a, b);
```

関数テンプレートの型名でないパラメータ

新リリースでは、次のような関数テンプレートの型名でないパラメータをサポートします。

```
template<int I> void foo( int a[I] ) { ... }
template<int I> void foo( mytype<I> m ) { ... }
```

新リリースでは、次の例にあるように、型名でないテンプレートパラメータが入った式を関数パラメータリストに使用することはできません。

```
// 以下はサポートされません
template<int I> void foo( mytype<2*I> ) { ... }
template<int I, int J> void foo( int a[I+J] ) { ... }
```

メンバーテンプレート

標準モードでのクラスとクラステンプレートには、次のコード例のように、メンバーとしてテンプレートを持たせることができます。

```
template <class T1>
class OuterClass {
public:
    // クラスメンバーテンプレート
    template <class T2>
        class MemberClass
    {
       T2 MCmember;
       T1 OCmember;
    };
    template<class T3> operator T3() { ... }
...
};
```

注 – メンバーテンプレートは、互換モード (`-compat [=4]`) ではサポートされていません。

定義分離テンプレート編成に対する制約の解除

新リリースのコンパイラでは、-instances != extern (-instances=explicit、-instances=global、-instances=semiexplicit、または -instances=static) における「定義分離テンプレート編成」に対する制約が解除されています 。-instances の設定値にかかわりなく、定義検索時に個別のソースファイルを含めるのがコンパイラのデフォルトの動作となっています。

この制約を再び有効にするには、-template=no%extdef オプションを使用してください。ただし、-template=no%extdef オプションが指定されている場合、コンパイラは -instances=extern が指定されても個別のソースファイルを検索しません。

静的変数破棄の順序付け

規格により、静的記憶期間のあるオブジェクトを破棄する順序がさらに詳しく定義されました。静的オブジェクトは、生成した順序とは逆の順序で破棄する必要があります。以前の言語の定義では、いくつかの局面での規定が放置されていました。

厳しくなったこの順序付けは標準モードの場合に限り実装されます。互換モード (-compat[=4]) では、破棄の順序は以前と同様に実装されます。

プログラムが破棄について特定の順序に依存しており、しかも旧コンパイラで動作していた場合、標準モードでは、この規格で要求される順序によってプログラムが破壊されてしまう可能性があります。この場合、-features=no%strictdestrorder コマンドオプションを使用すると、破棄の厳格な順序付けが行われなくなります。

副集合の初期設定

(中括弧初期設定を使用できる型に関し) クラスオブジェクトの中括弧初期設定を使用している場合、C++ 標準では、それ自体が集合クラスであるメンバーは、自分自身の型の値で初期設定できるようになりました。以下のコード例を参照してください。

```
struct S {  // 集合の型
int i, j;
};
struct T { // 集合の型
S s; // 集合のメンバー
int k;
};
T t1 = { {1, 2}, 3 }; // 従来の初期化
S s1 = { 1, 2 };
T t2 = { s1, 3 }; // 副集合の初期設定
```

自分専用の C++ 標準ライブラリの使い方

コンパイラ付属のライブラリの代わりに、自分専用の C++ 標準ライブラリを使用する場合は、`-library=no%Cstd` オプションを指定します。このオプションは、以下のヘッダーを検索しないようにします。

`<algorithm> <bitset> <complex> <deque> <fstream> <functional> <iomanip> <ios> <iosfwd> <iostream> <istream> <iterator> <limits> <list> <locale> <map> <memory> <numeric> <ostream> <queue> <set> <sstream> <stack> <stdexcept> <streambuf> <string> <strstream> <utility> <valarray> <vector>`

`-library=no%Cstd` を指定すると、前述のヘッダーを実装する `libCstd` ライブラリがプログラムに自動的にリンクされなくなります。上記のヘッダーで宣言されている機能を使用するには、`-I` オプションを使用して代わりに使用するヘッダーのあるディレクトリを指定し、プログラムを、代替ヘッダーの実装を含むライブラリまたは一連のオブジェクトファイルとリンクする必要があります。

上に記載したヘッダーの一部だけを入れ替えたり、`libCstd` を、別のライブラリの実装の全部または一部とリンクすることは安全ではありません。たとえば、文字列クラスだけを置き換えて `libCstd` を使用することはできません。コンパイラ付属のライブラリを使用するか、または上記に記載したすべての機能を置き換えてください。

残りのヘッダー (<exception>、<new>、<typeinfo>、および C から継承したすべてのヘッダー) は、コンパイラ自体または Solaris にとって必要不可欠で、-library=no%Cstd オプションでその使用の有無を制御することはできません。また、ライブラリ libCrun のリンクも -library=no%Cstd オプションでは制御できません。

libCrun の機能を置き換えることはできません。標準ライブラリを置き換える場合は、コードをコンパイラ付属の <exception>、<new>、および <typeinfo> と共にコンパイルする必要があります。標準モード (デフォルトモード) では、C++ プログラムは必ず libCrun にリンクしてください。

注 – このオプションは、自分の責任において使用してください。自分専用のバージョンの C++ 標準ライブラリを使用しても、最適な結果が得られないことがあります。

キャッシュバージョン指定

C++ コンパイラはキャッシュバージョンの違いを検出し、適切なエラーメッセージを出す機能があります。コンパイラでは、それぞれのテンプレートキャッシュディレクトリにテンプレートのキャッシュバージョンを固有に識別するバージョンを示す文字列を付けます。コンパイラの後続のリリースもキャッシュのバージョンを示す文字列を使用しますが、これらのバージョンは現行のバージョンと違うことがあります。

このコンパイラと後続のコンパイラはキャッシュディレクトリ内からバージョンストリングを検出して、適宜エラーを出します。たとえば、後続のコンパイラは異なるテンプレートキャッシュバージョンを使用し、このコンパイラのリリースによって生成されるキャッシュディレクトリを処理しようとすると以下のエラーを出力する場合があります。

```
SunWS_cache: Error: Database version mismatch
/SunWS_cache/CC_version
```

同様に、このリリースのコンパイラでは、将来リリースされるコンパイラによって生成されたキャッシュディレクトリを検出するとエラーを出力します。

Sun WorkShop C++ コンパイラ 5.0 によって生成されるテンプレートキャッシュディレクトリにはバージョン指定が行われていません。ただし、Sun WorkShop 6 C++ コンパイラでは、エラーや警告が出されることなくキャッシュディレクトリが処理されます。これらのキャッシュディレクトリは、Sun WorkShop 6 C++ コンパイラによって使用されるキャッシュディレクトリ形式に変換されます。

Sun WorkShop 6 C++ コンパイラまたはそれ以降のリリースによって生成されるテンプレートキャッシュディレクトリは、Sun WorkShop C++ コンパイラ 5.0 では使用できません。Sun WorkShop C++ コンパイラ 5.0 は、形式の違いを認識できないので、エラーを出力します。

Fortran コンパイラ

Sun WorkShop 6 には、Sun WorkShop™ Fortran コンパイラ `f77` と Sun WorkShop™ Fortran コンパイラ `f95` が含まれます。

この Sun WorkShop 6 リリースの Fortran コンパイラでサポートしている Solaris SPARC™ プラットフォーム版オペレーティング環境のバージョンは、2.6、7、および 8 に限定されます。Solaris Intel IA-32 プラットフォームについては、サンは Fortran のコンパイラと Sun Performance Library の開発を中止しました。このリリースでは、サンは Solaris Intel IA-32 プラットフォーム用の Forte™ for High Performance Computing または Forte™ Fortran デスクトップ版 (かつての Sun Performance WorkShop Fortran) を提供していません。Solaris Intel IA-32 プラットフォームのソフトウェア開発支援ツールの製品ラインについては、Portland Group (`http://www.pgroup.com`) に問い合わせてください。

Fortran 77 コンパイラ

表 3-3 は、Sun WorkShop 6 Fortran 77 コンパイラの新機能です。一部の機能について詳細は、73 ページの「Fortran コンパイラの新機能」を参照してください。

表 3-3　FORTRAN 77 コンパイラの新機能

機能	説明
入出力操作に対する FORM="BINARY" の効果	この新しいオプションを `OPEN(..)` 文の中で指定すると、レコードマークなしの順番探査ファイル (書式なし) としてファイルが取り扱われます。この結果、連続したバイトストリームとしてデータが読み書きされ、他のベンダーシステムとの互換性が確保されます。この機能は、Fortran 95 と Fortran 77 の両方のコンパイラに実装されています。
最適化コードのデバッグ	新リリースでは `-g` を使用したコンパイルに対する制約が緩和され、デバッグ用 (`-g`) のフラグとともに `-O4` と `-O5` でのコンパイル、またはすべての並列化フラグ (`-parallel`、`-explicitpar`、`-autopar`) によるコンパイルが可能になりました。
新しいコマンド行フラグ	新リリースの `f77` には、次のコマンド行フラグが追加されています (`f77`(1) マニュアルページ参照)。 • `-aligncommon` – 指定のバイト境界に共通ブロック要素を整列します。 • `-r8const` - 単精度データ定数を `REAL*8` に拡張します。 • `-xmemalign` – データ要素の一般的整列を指定します。

表 3-3　FORTRAN 77 コンパイラの新機能 (続き)

機能	説明
拡張コマンド行フラグ	次の `f77` コマンド行フラグが拡張されました (`f77`(1) マニュアルページ参照): • `-fast` – `-O5`、`-fsimple=2`、`-xvector=yes`、`-pad=common` を設定します。 • `-xprefetch` – 明示的なプラグマ `prefetch` 命令を使用可能にし、UltraSPARC プラットフォームで先読み命令を生成します。 • `-xtypemap` – データ型のサイズを指定することができます。
Cray スタイルの指令[1]	Cray スタイルの並列化指令に AUTOSCOPE を追加。
ハイパーリンクされたコンパイラの診断	Sun WorkShop の「構築」ウィンドウの `f77` のエラーメッセージには、メッセージを説明するヘルプページへのハイパーリンクが追加されました。

1) ライセンス供与 Fortran 77 コンパイラの並列化機能には、Sun WorkShop HPC ライセンスが必要です。

Fortran 95 コンパイラ

表 3-4 は、Sun WorkShop 6 Fortran 95 コンパイラの新機能です。一部の機能について詳細は、73 ページの「Fortran コンパイラの新機能」を参照してください。

表 3-4　Fortran 95 コンパイラの新機能

機能	説明
標準への準拠	Fortran 95 コンパイラは、Fortran 95 の規格に完全に準拠しています。
新しいコマンド	Fortran 95 コンパイラは、`f90` と `f95` のどちらのコマンドによっても起動できます。`f95` コマンドは新しいコマンドです。`f90` は `f95` と同等のコマンドです。
ファイル拡張子	コンパイラは、拡張子 `.f90` と `.F90` のほか、`.f95` と `.F95` のソースファイルも受け付けます。

表 3-4　Fortran 95 コンパイラの新機能 (続き)

機能	説明	
入出力操作に対する FORM="BINARY" の効果	この新オプションを OPEN(..) 文の中で指定すると、レコードマークなしの順番探査ファイル (書式なし) としてファイルが取り扱われます。この結果、連続したバイトストリームとしてデータが読み書きされ、他のベンダーシステムとの互換性が確保されます。この機能は、Fortran 95 と Fortran 77 の両方のコンパイラで実装されています。『FORTRAN 77 言語リファレンス』を参照してください。	
最適化コードのデバッグ	-g を使用したコンパイルに対する制約が緩和され、デバッグ用 (-g) のフラグとともに -O4 と -O5 でのコンパイルや、すべての並列化フラグ (-parallel、-explicitpar、-autopar) によるコンパイルが可能になりました。	
F77 フラグ	f77 コンパイラフラグのほとんどは、f95 で実装されています。詳細については、f95(1) マニュアルページを参照してください。具体的なフラグは、次のとおりです。	
	-erroff	指定したエラーメッセージを非表示にします。
	-errtags	タグ付きのエラーメッセージを表示します。
	-ext_names	下線付きまたは下線なしの外部名を作成します。
	-fpp	ソースコードプリプロセッサを指定します。
	-loopinfo	どのループが並列化されているかを示します。
	-sbfast	ブラウザテーブル情報を作成します。
	-silent	コンパイラメッセージを抑止します。
	-U	小文字と大文字の区別を許可します。
	-u	IMPLICIT NONE を意味します。
	-xcrossfile	ファイル間の最適化を使用可能にします。
	-xF	アナライザのための関数レベルの順序変更を可能にします。
	-xinline	関数をインラインでコンパイルします。
	-xtypemap	デフォルトのデータサイズを指定します。
新しいフラグ	次の新しいフラグが f95 で実装されています。	
	-aligncommon	指定のバイト境界に共通ブロック要素を整列します。

表 3-4　Fortran 95 コンパイラの新機能 (続き)

機能	説明	
	-mp=openmp	OpenMP 指令を受け付けます。
	-r8const	単精度定数を REAL*8 に高めます。
	-xia	区間演算拡張機能の処理を使用可能にします。(推奨)
	-xinterval	区間演算拡張機能の処理を使用可能にします。
	-xmemalign	データ要素の適切な境界への整列を指定します。
	-xrecursive	RECURSIVE 属性なしの再帰的呼び出しを許可します。
拡張フラグ	以下の f95 コマンド行フラグが拡張されました (f95(1) マニュアルページ参照)。	
	-fast	-05 -fsimple=2 -xvector=yes -pad=common を設定します。
	-xprefetch	明示的なプラグマ prefetch 指令を使用可能にし、UltraSPARC プラットフォームで先読み命令を生成します。
	-xtypemap	可能なデータ型指定の拡張セットを含んでいます。
OpenMP	このリリースの Fortran 95 は、ソースコード指令のセット、実行時ライブラリルーチン、環境変数を含む、明示的並列化のための OpenMP インタフェースを実装しています。『Fortran ユーザーズガイド』を参照してください。	
Cray スタイルの指令	Cray スタイルの並列化指令に追加された AUTOSCOPE。	
区間演算拡張機能	このリリースの Fortran 95 では、組み込みの INTERVAL データ型のサポートが実装されています。	
ライセンス	Fortran 95 コンパイラの並列化機能には、Sun WorkShop HPC ライセンスが必要です。	
ハイパーリンクされたコンパイラの診断	Sun WorkShop 「構築」ウィンドウでは、f95 のエラーメッセージをオンラインヘルプへのリンクとして解釈されるようになりました。	

Fortran コンパイラの新機能

以下の節では、Fortran コンパイラのいくつかの新機能についてさらに詳しく説明します。

入出力操作に対する FORM="BINARY" の効果

- WRITE 文 − データはバイナリとしてファイルに書き込まれ、出力リストに指定されているバイト数が転送されます。
- READ 文 − データは入力リスト上の変数に読み込まれ、同リストで要求されているバイト数が転送されます。ファイルにはレコードマークがないので、「記録終了」のエラー検出は行われません。検出対象のエラーは、「ファイル終了」やシステム異常エラーだけです。
- INQUIRE 文 − FORM="BINARY" で開いたファイルに対する INQUIRE は、以下を返します。

  ```
  FORM="BINARY"
  ACCESS="SEQUENTIAL"
  SEQUENTIAL="YES"
  DIRECT="NO"
  FORMATTED="NO"
  UNFORMATTED="YES"
  RECL= と NEXTREC= は未定義。
  ```

- BACKSPACE 文 − 使用は許可されていないので、エラーが出力されます。
- ENDFILE 文 − 通常どおり、現在位置でファイルを切り捨てます。
- REWIND 文 − 通常どおり、データの先頭にファイルを再配置します。

OpenMP

このリリースの Fortran 95 は、ソースコード指令のセット、実行時ライブラリルーチン、環境変数などをはじめとする、明示的並列化のための OpenMP インタフェースを実装しています。OpenMP の暫定ドキュメントが README として入っています。OpenMP の仕様については、http://www.openmp.org/ を参照してください。

Fortran コンパイラで受け付ける OpenMP を含むすべての指令については、『Fortran ユーザーズガイド』の付録 E にまとめられています。Fortran コンパイラの並列化機能について詳しくは、『Fortran プログラミングガイド』を参照してください。

注 - Fortran のコンパイラの並列化機能には、Sun WorkShop HPC ライセンスが必要です。

区間演算拡張機能

このリリースの Fortran 95 には、区間演算拡張機能が実装されています。74 ページの「Fortran 95 区間演算」を参照してください。

ハイパーリンクされたコンパイラの診断

Sun WorkShop を使用してアプリケーションの構築やコンパイルを行う場合、「構築」ウィンドウ内の `f77` と `f95` の診断メッセージにヘルプページへのハイパーリンクが追加されました。このエラーメッセージをクリックすると、ヘルプブラウザが立ち上がり、特定のエラー診断に関する詳しい情報が表示されます。

Fortran 95 区間演算

組み込み `INTERVAL` データ型のサポートは、Sun WorkShop 6 Fortran 95 コンパイラの新しい機能です。

新しい 2 つのコンパイラフラグ `-xia` と `-xinterval` により、コンパイラは区間特有の言語を認識し、区間命令を実現するためのコードを生成します。

区間演算とは

区間演算を使用すると、各区間に含まれる数の集合について算術演算の値が求められます。区間とは、その区間の下位の境界から上位の境界までの間にあるすべての実数の集合を意味します。区間演算結果はどれもすべての可能な結果の集合が必ず含まれるひとつの新しい区間です。

Sun WorkShop 6 Fortran 95 を使用すれば、区間プログラムを作成して算術演算の値で厳密な境界を計算するのは容易です。

- 変数を `INTERVAL` 型として宣言します。

- INTERVAL 組み込み関数や演算子、関係演算子、および書式編集記述子を使用して通常の Fortran コードを作成します。
- -xia コマンド行オプションを使用してコードをコンパイルします。

最適な結果を得るためには、狭幅の区間結果を計算する既存の区間アルゴリズムを使用します。そして、この狭幅の区間結果を計算するアルゴリズムを考案することが、区間分析の主な目的です。

区間演算が重要である理由

区間演算は次の理由により重要です。

- 区間演算を使用すると、入力データエラー、マシンの丸め、それらの相互作用など、あらゆるソースからのエラーに確実な範囲を設けてコンピュータで計算を実行できます。
- 区間アルゴリズムにより、非線形的な問題の解決、たとえば、方程式の非線形システムや非線形プログラミングを行なうことが可能になります。

区間がより一般的になるにつれて、区間ソルバーのライブラリが線形および非線形を問わずあらゆる問題に対するシャープな区間解決法の計算に利用できるようになり、さらにエラーのあらゆる原因を考慮できるようになります。これらのライブラリを用いることによって、科学者やエンジニア、商用アプリケーションの開発者は、それまでは手の届かなかった問題を解決するプログラムを開発できるようになります。

詳細情報の入手先

『Fortran 95 区間演算プログラミングリファレンス』または区間演算の README に記載されているオンラインリソースの一覧を参照してください。

dbx

表 3-5 は、Sun WorkShop 6 dbx の新機能です。

表 3-5　dbx の新機能

機能	説明
firedhandlers ksh 変数	読み取り専用 ksh 変数 firedhandlers が追加されています。この変数を delete コマンドと handler コマンドと併用すれば、clear コマンドの代わりとして使用できます。詳細は、Sun WorkShop オンラインヘルプの「dbx コマンドの使い方」の「変数」、「delete コマンド」、「handler コマンド」を参照してください。
ブレークポイントの部分的消去	新リリースの clear コマンドによって、クラス内、メソッド内、関数内のブレークポイントを部分的に消去することが容易になります。詳細は、Sun WorkShop オンラインヘルプの「dbx コマンドの使い方」の 「clear コマンド」を参照してください。
トレース出力	トレースの出力をファイルにリダイレクトできます。詳細は、Sun WorkShop オンラインヘルプの「dbx コマンドの使い方」の 「trace コマンド」を参照してください。

表 3-5　dbx の新機能 (続き)

機能	説明
新しい dbx 環境変数	• 新しい dbx 環境変数 stack_find_source は、プログラム停止時にデバッグ可能ソースコードが含まれているフレームまでコールスタックを dbx が自動的に上方移動させるかどうかを制御します。詳細は、Sun WorkShop オンラインヘルプの「dbx コマンドの使い方」の 「stack_find_source 環境変数」を参照してください。 • 新しい dbx 環境変数 proc_exclusive_attach は、別のデバッガやデバッグツールの制御下にあるプロセスに dbx を接続できるかどうかを制御します。詳細は、Sun WorkShop オンラインヘルプの「dbx コマンドの使い方」の 「proc_exclusive_attach 環境変数」を参照してください。 • 新しい dbx 環境変数 step_granularity は、step コマンドと next コマンドが文、行、ソースコードのどれを処理対象とするかを制御します。詳細は、Sun WorkShop オンラインヘルプの「dbx コマンドの使い方」の 「step_granularity 環境変数」を参照してください。 • 新しい dbx 環境変数 mt_scalable は、資源の使用量を削減することによって、多くの LWP (軽量プロセス) を使うマルチスレッドアプリケーションのデバッグを行いやすくします。詳細は、Sun WorkShop オンラインヘルプの「dbx コマンドの使い方」の 「mt_scalable 環境変数」を参照してください。 • 新しい dbx 環境変数 rtc_error_stack は、スタックトレースが RTC 内部構造に対応するフレームを表示するかどうかを決定します。詳細は、Sun WorkShop オンラインヘルプの「dbx コマンドの使い方」の 「rtc_error_stack 環境変数」を参照してください。

表 3-5　dbx の新機能 (続き)

機能	説明
	• 新しい dbx 環境変数 rtc_inherit は、デバッグプログラムから実行されている子プロセスで実行時検査を行うかどうかを決定します。詳細は、Sun WorkShop オンラインヘルプの「dbx コマンドの使い方」の「rtc_inherit 環境変数」を参照してください。 • 新しい dbx 環境変数 rtc_use_traps は、実行時チェックでの 8 メガバイトコード制限の回避を有効にします。詳細は、Sun WorkShop オンラインヘルプの「dbx コマンドの使い方」の「rtc_use_traps 環境変数」を参照してください。
C++ メンバー関数におけるブレークポイント	C++ インラインメンバー関数のブレークポイントは、正しく動作します。ハンドラ条件 (-if) も正しく評価されます。
LWP 関連コマンド	このリリースでは、マルチスレッドアプリケーションのデバッグ時だけではなく、いつでも LWP 関連のコマンドを利用できます。詳細は、Sun WorkShop オンラインヘルプの「dbx コマンドの使用」の「lwp コマンド」と「lwps コマンド」を参照してください。
実行中のプロセスの中断	Control-C を押してもハングアップしたプロセスが停止しない場合は、3 回連続して Control-C を押すと、ハングアップしたプロセスを強制的に停止できます。Sun WorkShop オンラインヘルプの「dbx コマンドの使い方」の「実行中のプロセスの中断」を参照してください。
全行番号情報	-g オプションと -o オプションによるコンパイル時に、全行番号情報が記録されるようになりました。これで、step コマンドと next コマンドを使用して、最適化コードを順次実行していくことができます。ただし、現行の行は、コードスケジューリングの都合上、順方向と逆方向にジャンプします。dbx 内から出力される変数値は、最適化コードでは引き続き信頼に欠けます。

Sun WorkShop 6

表 3-6 は、Sun WorkShop 6 の新機能です。一部の製品の機能については、表の次の節でさらに詳しく説明しています。

表 3-6 Sun WorkShop 6 の新機能

機能	説明
テキスト編集	
式のバルーン評価	式のバルーン評価は、テキストエディタで現在マウスでポイントしている数式の現在値をその場で表示します。数式のタイプを調べたり、ポインタを間接参照したりすることもできます。
NEdit と Vim テキストエディタ	今回のリリースでは NEdit と Vim が統合型テキストエディタとして加わりました (英語環境でのみ使用できます。日本語環境では使用できません)。
プログラムのデバッグ	
「ボタン編集」ウィンドウ	「ボタン編集」ウィンドウにより、次の処理が行えます。 エディタウィンドウと「デバッグ」ウィンドウのツールバーのカスタマイズ。「デバッグ用カスタムボタン」ウィンドウのボタンの追加、削除、または編集。
デバッグウィンドウ	「データ履歴」区画が、「データ履歴」タブ、「プログラム入出力」タブ、それにオプションの「データ表示」タブと置き換えられました。 「セッション」区画と「スレッド」区画を切り替えるためのラジオボタンが、「セッション」タブ、「スレッド」タブ、および「ブレークポイント」タブと置き換えられました。
「デバッグオプション」ダイアログボックス	オプションのカテゴリを切り替える「カテゴリ」リストボックスは、ダイアログボックスの右側のタブに置き換えられています。

表 3-6　Sun WorkShop 6 の新機能 (続き)

機能	説明
新しいデバッグオプション	「デバッグ」ウィンドウの「プログラム入出力」タブにプログラムへの入出力指示に関する選択肢が追加されました。 デバッグ可能なソースコードではないプログラム内の関数で実行が停止した場合に、コールスタックを最初のデバッグ可能ソースコードまでポップするよう選択できるようになりました。 実行単位を「行」に設定するよう選択できるようになったため、含まれている文の数に関係なく、次の 1 コマンドを特定の 1 行に対して実行できます。 デフォルトの設定では、ツールバーの下に「状態」、「停止位置」、「評価コンテキスト」、セッション数などの情報は表示されなくなりましたが、従来どおり表示されるよう選択することもできます。 プログラムの `main()` モジュールがデバッグ情報付きでコンパイルされない場合でも、デバッガが警告を出させないよう選択できるようになりました。 「データ表示」ウィンドウを「デバッグ」ウィンドウのタブとして、または別のウィンドウとして表示するよう選択できるようになりました。 `-xs` フラグ付きでコンパイルされたモジュールをデバッグセッションの開始時に読み込むのではなく、それらのモジュールのデバッグ情報が必要となった時点で読み込むよう選択できるようになりました。 多数 (300 以上) の LWP (軽量プロセス) を使用するプロセスをデバッグする必要がある場合、デバッグ時にリソースの使用量が少なくなるよう設定できます。 デバッガに `dbx` がデバッグ対象プログラムを排他的に制御しているかどうかをチェックさせないよう選択できます。 デバッガが、`dbx` の機能にとって欠かせない一定の共有ライブラリを除外できるようにすることが可能になりました。
削除されたデバッグオプション	「マルチスレッド使用のプログラムのウォッチポイントを許可」のオプションが「デバッグオプション」ダイアログから削除されました。
プロジェクトの取り扱い	

表 3-6　Sun WorkShop 6 の新機能 (続き)

機能	説明
プロジェクトと ワークセット	この Sun WorkShop のリリースでは、プロジェクトを使って開発プロジェクトに関連付けられたファイル、プログラム、およびターゲットを追跡し、メークファイルを作成しなくてもプログラムを構築できます。
新しいマニュアルページ	
`makeprd`(1)	Sun WorkShop プロジェクトファイルビルダ
`nedit`(1)	Motif UI スタイルのテキストエディタ (英語のみ)
`vim`(1)	vi の改良版といえるプログラマ用テキストエディタ (英語のみ)
`xemacs`(1)	次世代 Emacs

テキスト編集

以下の 2 つの節は、新しく追加された Sun WorkShop 6 テキスト編集機能を説明しています。

式のバルーン評価

式のバルーン評価は、テキストエディタにおいて現時点でマウスでポイントしている数式の現在値を即時に表示します。数式のタイプを調べたり、ポインタを間接参照したりすることもできます。詳細については、オンラインヘルプの「テキスト編集」の「式のバルーン評価の使用」を参照してください。

NEdit と Vim テキストエディタ

テキストエディタは、構築、デバッグ、ブラウズを含む Sun WorkShop 統合開発支援ツールセットの中心に位置します。Sun WorkShop プログラミング環境では、数式の評価、ブレークポイントの設定、関数の検討をテキストエディタから行うことができます。

X/Motif システムへのグラフィカルユーザーインタフェースを備えたプレーンテキストエディタである NEdit、および UNIX システムでの改良版 vi 標準テキストエディタである Vim は、このリリースでの新しい統合テキストエディタです。以下は、Sun WorkShop 6 とともに提供される統合エディタのリストです。

- NEdit
- XEmacs
- GNU Emacs
- Vi
- Vim (グラフィカルユーザーインタフェースオプションあり)

各エディタの選び方の詳細については、以下を参照してください。

- エディタのメニューバーのヘルプメニューで利用できるオンラインマニュアル
- オンラインヘルプの「テキスト編集」の「テキストエディタオプションダイアログ」

プログラムのデバッグ

以下の節は、新しく追加されたデバッグ機能について説明しています。

「ボタン編集」ウィンドウ

「ボタン編集」ウィンドウにより、エディタウィンドウと「デバッグ」ウィンドウのツールバーをカスタマイズしたり、「デバッグ用カスタムボタン」ウィンドウのボタンを追加、削除、または編集したりできるようになりました。

詳細については、オンラインヘルプの「デバックウィンドウの使い方」の「ツールバーオプション」を参照してください。

「デバッグ」ウィンドウ

以下の節では、「デバッグ」ウィンドウに組み込まれたいくつかの変更について説明しています。

セッションステータスとコンテキスト情報

デフォルトの設定では、「状態」、「停止位置」、「評価コンテキスト」、「セッション数」などの情報は、これまでのようにはツールバーの下に表示されません。この情報を表示させたい場合は、「デバッグオプション」ダイアログの「ウィンドウレイアウト」タブの中の「画面の上部にコンテキスト / ステータスの 3 行を表示」を選択します。詳細については、オンラインヘルプの「デバッグウィンドウの使い方」の「セッションステータス情報とコンテキスト情報の表示」を参照してください。

「データ」区画

「データ」区画は、「データ履歴」と「プログラム入出力」のタブ、およびオプションの「データ表示」タブと置き換えられました。

- 「データ履歴」タブには、「データ」区画が表示されます。詳細については、オンラインヘルプの「デバッグウィンドウの使い方」の「「データ履歴」タブ」を参照してください。
- 「プログラム入出力」タブは、別の「プログラム入出力」ウィンドウではなく、「デバッグ」ウィンドウ内にプログラムの入出力に関する情報を表示します (詳細は、オンラインヘルプの「デバッグウィンドウの使い方」の「「プログラム入出力」タブ」を参照してください)。デフォルトではこのタブにプログラムの入出力情報が表示されるようになりましたが、「デバッグオプション」ダイアログボックスの「デバッグ出力」タブの「プログラム出力」区画を使用して、プログラムの入出力を行う場所を選択できます。詳細については、オンラインヘルプの「デバッグウィンドウの使い方」の「プログラムの入出力先のリダイレクト」を参照してください。
- 「データ表示」タブは、「デバッグオプション」ダイアログボックスの「データ表示ウィンドウ」タブの「データ表示ウィンドウ 以下として表示」区画で「デバッグウィンドウのタブ」を選択した場合に (別の「データ表示」ウィンドウの代わりに) 含まれます。詳細については、オンラインヘルプの「デバッグウィンドウの使い方」の「「データ表示」タブ」と「「データ表示」ウィンドウの表示方法の選択」を参照してください。

セッション区画とスレッド区画

「セッション」区画と「スレッド」区画を切り替えるためのラジオボタンが、「セッション」、「スレッド」、および「ブレークポイント」のタブと置き換えられました。

- 「セッション」タブは「セッション」区画を表示します。このタブの上で右マウスボタンを押すと、ポップアップメニューが表示されます。「セッションタブ」を参照してください。詳細については、オンラインヘルプの「デバッグウィンドウの使い方」の「「セッション」タブ」を参照してください。
- 「スレッド」タブは「スレッド」区画を表示します。詳細については、オンラインヘルプの「デバッグウィンドウの使い方」の「「スレッド」タブ」を参照してください。

- 「ブレークポイント」タブは、プログラムに割り当てられているブレークポイントやトレースポイントをスクロール可能な一覧として表示します。このタブの上で右マウスボタンを押すと、ポップアップメニューが表示されるので、各ブレークポイントごとに瞬時に使用可能、使用禁止、削除、あるいはソースを表示することができます。このポップアップメニューには「追加」項目が含まれており、この項目を選択すると、ブレークポイントの追加、使用可能、使用禁止、変更、削除などの操作を行える別の「ブレークポイント」ウィンドウが表示されます。詳細については、オンラインヘルプの「デバッグウィンドウの使い方」の「「ブレークポイント」タブ」を参照してください。

「デバッグオプション」ダイアログボックス

「デバッグオプション」ダイアログボックスには、以下の新機能が追加されました。

「カテゴリ」タブ

「デバッグオプション」ダイアログボックスの「カテゴリ」リストボックスは、オプションのカテゴリを切り替えるためのウィンドウの右側に表示されるタブと置き換えられました。

新しい「プログラム出力」オプション

プログラムの入出力を、「デバッグ」ウィンドウの「プログラム入出力」タブに変更する選択肢が追加されました。以前と同様、別の「プログラム入出力」ウィンドウ、「dbx コマンド」ウィンドウ、あるいはカスタム `pty` にプログラムの入出力先を変更することもできます。詳細については、オンラインヘルプの「デバックウィンドウの使い方」の「プログラムの入出力先のリダイレクト」を参照してください。

新しい「コールスタック」オプション

デバッグセッション中、プログラムが `-g` フラグ付きでコンパイルされなかった場合、デバッグ可能なソースコードでないプログラム内の関数で実行が停止する場合があります。このバージョンでは、コールスタックをこのケースで最初のデバッグ可能なソースコードにポップするよう選択できるようになりました。詳細については、オンラインヘルプの「デバッグウィンドウの使い方」の「実行停止時にスタックをさかのぼる」を参照してください。

新しい「ステップ実行」オプション

デフォルトの設定では、デバッグのステップ実行単位は「文」に設定されます。したがって、1 つのソースコード行に複数の文が含まれている場合、その行をステップ実行するのにその文の数だけ next コマンドが必要です。新バージョンでは、ステップ実行単位に「行」を設定できるようになったため、行に含まれている文の数に関係なく、next コマンド 1 つで行全体をステップ実行できます。詳細については、オンラインヘルプの「デバックウィンドウの使い方」の「ステップ実行単位の設定」を参照してください。

新しい「ウィンドウレイアウト」オプション

デフォルトの設定では、「状態」、「停止位置」、「評価コンテキスト」、および「セッション数」の情報は、ツールバーの下に表示されなくなりました。これらの情報を表示させたい場合には、「デバッグオプション」ダイアログボックスの「ウィンドウレイアウト」タブで「画面の上部にコンテキスト / ステータスの 3 行を表示」を選択します。詳細については、オンラインヘルプの「デバックウィンドウの使い方」の「セッションステータス情報とコンテキスト情報の表示」を参照してください。

新しい「ウィンドウ動作」オプション

デフォルトの設定では、デバッガはプログラムの main() モジュールにデバッグ情報が付随していない場合に警告を出します。新バージョンでは、この警告を表示させないようにすることができます。詳細については、オンラインヘルプの「デバックウィンドウの使い方」の「デバッグ情報を使用しないで main() モジュールをコンパイルした時に警告を表示させる」を参照してください。

新しい「データ表示」オプション

「データ表示」ウィンドウを「デバッグ」ウィンドウのタブとして表示させたり、別のウィンドウとして表示させることができるようになりました。詳細については、オンラインヘルプの「デバックウィンドウの使い方」の「「データ表示」ウィンドウの表示方法」を参照してください。

新しい「デバッグパフォーマンス」オプション

-xs フラグ付きでコンパイルされたモジュールを、デバッグセッションの開始時に読み込むのではなく、それらのモジュールのデバッグ情報が必要となった時点で読み込むよう選択できるようになりました。このオプションの追加により、モジュールが

-xs フラグ付きでコンパイルされたときのデバッグ開始時間を短縮することができます。デフォルトでは、このデバッグオプションはオンです。詳細については、オンラインヘルプの「デバックウィンドウの使い方」の「-xs を使ってコンパイルしたモジュールの読み込みを遅延する」を参照してください。

新しい「フォークおよびスレッド」オプション

多数 (最大 300) の LWP (軽量プロセス) を伴うプロセスをデバッグする場合、新しくデバッグ時のリソース使用量を削減するよう設定できるようになりました。ただし、それによってデバッグのパフォーマンスは低下することがあります。詳細については、オンラインヘルプの「デバックウィンドウの使い方」の「大量の LWP のデバッグ」を参照してください。

新しい拡張オプション

デフォルトの設定では、デバッガは、dbx がデバッグ対象プログラムを排他的に制御するかどうかを検査します。このため、dbx は他のツールがすでのそのプログラムに接続されていると、そのプログラムに接続できません。このバージョンでは、この動作を解除することができるようになっています。詳細については、オンラインヘルプの「デバックウィンドウの使い方」の「プログラムの排他的制御が dbx にあることを確認する」を参照してください。

デフォルトでは、デバッガは dbx の機能にとって重要な一部の共有ライブラリの除外を許可していません。新バージョンでは、これらのライブラリを除外できるようになりましたが、その場合には中核のファイルしかデバッグできません。詳細については、ンラインヘルプの「デバックウィンドウの使い方」の「重要な dbx ライブラリの取り込みが必要」を参照してください。

プロジェクトの取り扱い

この Sun WorkShop のリリースでは、プロジェクトを使って開発プロジェクトに関連付けられたファイル、プログラム、およびターゲットを追跡し、メークファイルを作成しなくてもプログラムを構築することができます。プロジェクトは、実行プログラム、静的ライブラリまたは静的アーカイブ、共用ライブラリ、Fortran アプリケーション、複雑なアプリケーション、またはユーザーメークファイルアプリケーションの構築に使用されるファイルおよびコンパイラ、デバッガ、構築関連のオプションが含まれるリストです。

以前のバージョンの Sun WorkShop では、プロジェクトの代わりにワークセットが使用されていました。プロジェクトは依然としてワークセットの性質を持っていますが (プロジェクトに関連するファイルやディレクトリを簡単にアクセスできるようにするメニューピックリストは引き続き存在)、プロジェクトはユーザーがどのようなソースファイルをどのように作成したいか、ユーザーのプログラムをよく把握しています。Sun WorkShop のワークセットが手元にある場合は、それらを読み込み時に自動的に Sun WorkShop 6 のプロジェクトに変換できます。詳細については、オンラインヘルプの「プロジェクトの取り扱い」の「ワークセットをプロジェクトに変換する」を参照してください。

Sun WorkShop を起動すると、「Sun WorkShop にようこそ」ダイアログボックスが開き、すぐに Sun WorkShop プロジェクトとプロジェクトウィザードにアクセスできます。Sun WorkShop には、WorkShop のメインウィンドウの「プロジェクトメニュー」から利用できるプロジェクト機能があり、以下を実行できます。

- プロジェクトウィザードと自分専用のメークファイルまたは Sun WorkShop が作成してくれるメークファイルによる新しいプログラムの作成や単純なプログラムの構築 (オンラインヘルプの「プロジェクトの取り扱い」の「新規プロジェクトの作成」を参照)。
- 既存のプロジェクトの設定値の変更。設定値には、プロジェクトのコンパイル方法やソースブラウズ情報を生成させるかどうかの選択があります (オンラインヘルプの「プロジェクトの取り扱い」の「プロジェクトの編集」および「現在のプロジェクトの編集ウィンドウ」を参照)。

このほか、プロジェクトを読み込まない状態で Sun WorkShop 6 プログラミング環境を使用することも可能です。ピックリストは、開発プロジェクトに関連するファイル、プログラム、ディレクトリ、ターゲットを記録します (詳細については、オンラインヘルプの「プログラムの構築」の「WorkShop のターゲット」を参照)。各ファイルをアクセスし、ターゲットを構築し、実行形式ファイルをデバッグするといった操作を、WorkShop のメインウィンドウから行うことができます。構築ターゲット情報は常に一定ではないため編集できません。この情報は、構築ターゲットへアクセス、追加、削除するたびに変更されます。

Sun WorkShop TeamWare 6

表 3-7 は、Sun WorkShop TeamWare 6 の新機能です。一部の製品機能については、表の次の節でさらに詳しく説明しています。

表 3-7　Sun WorkShop TeamWare 6 の新機能

機能	説明
自動フリーズポイント機能	自動フリーズポイント機能は、特定のトランザクションの前または後にフリーズポイントファイルを自動的に作成します。フリーズポイントファイルの作成する時間をブリングオーバー、プットバック、取消し時、または衝突解決の前または後にするかを選択できます。
「バージョン管理」の「カスタマイズメニュー」	「カスタマイズメニュー」機能は、「バージョン管理」ウィンドウに「カスタマイズ」という新しいメニューを追加し、ユーザー固有のコマンドにアクセスできるようにします。カスタマイズメニューの作成については、オンラインヘルプの「管理ファイル」の「カスタマイズメニューの作成」を参照してください。
デルタコメント	この新オプションは、デルタ番号、所有者、コメントなどのデルタコメントをトランザクション出力と電子メール通知に追加します。「ワークスペース管理」で、「ワークスペース」/「作成ブリングオーバー」/「更新ブリングオーバー」/「プットバック」を選択して「デルタコメント」ボックスを 選択するか、または `-d` オプションを付けて `bringover`、`putback` コマンドを使用します。
ファイルマージの「相違ナビゲータ」	マージされていない 2 個のファイルの間に、「相違ナビゲータ」が表示されます。「相違ナビゲータ」の左右どちらかにあるスライドボックスをクリックすると 2 個のファイルのうち、どちらかがスクロールされ、最上部または最下部にある矢印をクリックすると両方のファイルにおいて同じ距離だけ移動します。
メニュー再編成の設定	Sun WorkShop TeamWare 6 では、ワークスペース属性のユーザーインタフェースが変更されています。

表 3-7　Sun WorkShop TeamWare 6 の新機能 (続き)

機能	説明
プットバックの妥当性検査	プットバックの妥当性検査をオンにすると、許可されたワークスペースだけがプットバックを行えるようになります。プットバックを行えるユーザーを管理し、特定のパスワードを要求するように設定することができます。
SCCS 管理フラグ	ファイルに対して SCCS 管理フラグを設定できます。
ワークスペース完全性検査	`workspace` コマンドで使用できる新オプション。`check [ -W ] [ -s ] wsname ...` は、ファイル、アクセスモード、親子関係、履歴ファイルの状態をチェックします。コマンドは、ワークスペースに問題がない場合は 0 を、エラーがある場合は 1 の終了コードで終了します。
ワークスペースの履歴表示	このリリースの Sun WorkShop TeamWare には、ワークスペース履歴ファイルに入っている情報を簡単に表示できる機能が含まれています。
ワークスペースのラベル	この機能により、チームメンバーがすぐに理解できるような、意味のある名前をワークスペースに指定できます。「ワークスペース」で「プロパティ」を選択して、「説明」タブを選びます。
新しいマニュアルページ	`description`(4)

メニュー再編成の設定

Sun WorkShop TeamWare 6 では、次のような変更がメニューに加えられています。

表 3-8　Sun WorkShop TeamWare 6 メニューの変更

TeamWare 2.2	TeamWare 6
ワークスペース管理:	
ファイル ➤ 親ワークスペースの読み込み	ワークスペース ➤親ワークスペースの読み込み
ファイル ➤ 子ワークスペースの読み込み	ワークスペース➤子ワークスペースの読み込み

表 3-8　Sun WorkShop TeamWare 6 メニューの変更 (続き)

TeamWare 2.2	TeamWare 6
ファイル ➤ 空の子ワークスペース作成	ワークスペース➤子ワークスペースの作成
編集 ➤削除	ワークスペース➤削除
編集 ➤名前変更	ワークスペース➤名前変更
編集 ➤親	ワークスペース➤親の変更
編集 ➤更新 ➤名前テーブル	ワークスペース ➤名前テーブル更新
トランザクション➤ ブリングオーバー ➤作成	アクション ➤作成ブリングオーバー
トランザクション➤ ブリングオーバー ➤更新	アクション ➤更新ブリングオーバー
オプション ➤ワークスペース	ワークスペース➤属性
オプション➤ワークスペース ➤ ロック編集	ワークスペース➤ロック編集
新規	表示 ➤再描画
新規	ワークスペース➤属性 ➤フリーズポイント
新規	ワークスペース➤属性 ➤ プットバックの妥当性検査
新規	オプション ➤ワークスペース管理 ➤子ワークス ペース読み込み: 選択する / すべて
バージョン管理	
新規	ファイル ➤ファイル情報
新規	コマンド ➤アンチェックアウト

プットバックの妥当性検査

プットバック妥当性検査をオンにすると許可されたワークスペースだけがプットバックを行えるようになります。プットバックを行う場合、「パスワード」(統合要求 ID：Integration Request Identifier) の入力を指示するプロンプトが表示されます。この機能は統合要求 ID を記録するだけで、チェックは行いません。統合要求 ID の

チェックを行うには、そのための妥当性検査プログラムを自分自身で作成する必要があります。詳細については、オンラインヘルプの「ワークスペースの管理」の「プットバックの妥当性検査」を参照してください。

SCCS 管理フラグ

SCCS 管理フラグを設定するには、「バージョン管理」で「ファイル」➤「ファイル情報」を選択します。たとえばプットバック中に「バージョン管理」に MR (修正要求文字列) の入力要求を出させるようにするには、妥当性検査プログラムの名前を「妥当性検査プログラム」テキストフィールドに指定してください。

ワークスペースの履歴表示

このリリースの Sun WorkShop TeamWare には、ワークスペース履歴ファイルに入っている情報を簡単に表示できる機能が含まれています (「ワークスペース」➤「履歴の表示」を選択)。ワークスペース履歴ビューアを利用すれば、ワークスペースのトランザクション履歴、トランザクション詳細、コメント、およびコマンドログを表示できます。エントリのソートやフィルタリングができるほか、コメントやコマンドログの検索も行えます。

Sun WorkShop Visual 6

表 3-9 では、Sun WorkShop Visual 6 で利用できる新機能を一覧表示します。一部の製品の機能については、表の次の節でさらに詳しく説明しています。

表 3-9　Sun WorkShop Visual 6

機能	説明
Swing サポート	新リリースの Visual 6 は、Java 1.0、1.1 に加え Java Swing コードも生成できます。Visual 6 は Motif ウィジェットの Swing コンポーネントを生成するばかりではなく、マップ可能資源の種類も増えています。画像、シェルアイコン、シェルのサイズ変更・削除応答、リスト内容、行/列エントリ位置合わせといった拡張機能を含むトグルの Swing コードを生成できるようになっています。
Windows 拡張サポート	Windows MFC への X イベントのマッピングサポートがさらに強化されています。
Sun WorkShop プロジェクトとの統合	Sun WorkShop プロジェクトウィザードと併用することによって、グラフィカルユーザーインタフェースを持つプロジェクトを作成できます。

Swing サポート

新リリースの Visual 6 は、Java 1.0、1.1 に加え Java Swing コードも生成できます。Visual 6 は Motif ウィジェットの Swing コンポーネントを生成するばかりではなく、マップ可能リソースの種類も増えています。画像、シェルアイコン、シェルサイズ変更、削除応答、リスト内容、行/列エントリ位置合わせといった拡張機能が入ったトグルの Swing コードを生成できるようになっています。

標準クラスに等価性が欠落している Java に Motif コンポーネントをマップする MWT クラスライブラリを Swing に移植することによって、見た目と使い心地の整合性が改善されています。一部の Motif コンポーネントは該当する Swing コンポーネントに直接マップされるようになっているので、MWT にそれほど依存しなくとも Motif 互換のインタフェースを用意できるようになっています。

Java Layout エミュレーションウィジェットに適宜手が加えられた結果、Java Layout 特性における動作の整合性が改善されています。

他社製 (Motif 以外) コンポーネントのクロスプラットフォームコードをサポートする動きの一環として、新リリースの Visual 6 ではあらゆる統合コンポーネントのデフォルトの基本クラス指定を行なうことができるようになっています。このクラス指定は、一般的言語ベースで行うことも、個別のバリアントについて行うこともできます。たとえば、Java、Java 1.0、1.1 のデフォルトクラスや Swing 固有のクラスを指定できます。コンポーネント MFC クラスを指定することもできます。ターゲット言語において正しい種類のオブジェクトを作成するには他社コンポーネントを 1 つ 1 つ手作業で設定しなければならなかった従来の問題が解決されています。他社固有のリソースはマップされません。特定のコンポーネントを複数のネイティブオブジェクトにマップしなければならない状況や、組み込み要素を考慮すれば複合コンポーネントをマップできる状況については、対応していません。

Windows 拡張サポート

Visual 6 には、X イベントを Windows MFC にマップするための次のサポートが追加されています。

表 3-10 X イベントを Windows MFC にマップする

MouseMotion (マウスの動き)	ボタンの押し下げの有無にかかわらず、あらゆるマウス移動について汎用ハンドラを生成します
ButtonPress	Left、Center、Right の 3 種類の押し下げハンドラすべてを生成します
ButtonRelease	Left、Center、Right の 3 種類の解放ハンドラすべてを生成します
EnterWindow	`MouseActivate`
ExposureMask	`EraseBkgnd`
KeyPressMask	`WM_KEYDOWN`
KeyRelease	`WM_KEYUP`
KeymapstateMask	`WM_SYSKEYUP/WM_SYSKEYDOWN`
LeaveWindowMask	`WM_KILL_FOCUS`
ResizeRedirect	`WM_SIZE`
PropertyChangeMask	`ON_WM_PAINT`
VisibilityChangeMask	`WM_SHOWWINDOW`

Sun Performance Library

Sun Performance Library™ は、線形代数およびその他数に集中した問題を解決するための最適化された高速数値演算のサブルーチンです。Sun Performance Library は、`http://www.netlib.org` にある Netlib で利用できるパブリックドメインアプリケーションのコレクションに基づいています。これらのルーチンは機能強化され 、バンドルされて Sun Performance Library となりました。

表 3-11 は、Sun Performance Library の Sun WorkShop 6 リリースで使用できる新機能です。一部の製品の機能については、表の次の節でさらに詳しく説明しています。

表 3-11 Sun Performance Library の新機能

機能	説明
LAPACK 3.0 のサポート	LAPACK 3.0 サブルーチンが追加されました。Sun Performance Library の以前のバージョンは、LAPACK 2.0 に基づいていました。 Sun Performance Library の現行バージョンでは、引き続き LAPACK 2.0 と LAPACK 1.X との互換性を保っています。
スパースソルバーパッケージ	スパースソルバーパッケージは、直接的な方法およびユーザー指定の順序付けを含むフィルリダクション順序付けのアルゴリズムを使用して、スパース行列 (対称的、構造上対称的、および非対称的な係数行列) を解決するルーチンを提供します。
UltraSPARC-III のサポート	UltraSPARC-III がサポートされるようになりました。UltraSPARC-III に固有のコードを使用するには、32 ビットコードの場合は `-xarch=v8plusb`、64 ビットコードの場合は `-xarch=v9b` のオプションを付けてコンパイルします。
Fortran 95 言語機能サポート	Fortran 95 言語の機能をサポートするようになりました。プログラムに F95 の文 `USE SUNPERF` を含めることによって、Performance Library のモジュールと定義を使用できます。
Sun Performance Library のライセンス供与に関する変更	Sun Performance Library は、ライセンスを必要としなくなりました。しかし、引き続き `-xlic_lib=sunperf` を使用して、アプリケーションを正しいサポートライブラリと確実にリンクし、正しいバージョンの Sun Performance Library を選択してください。

Fortran 95 言語機能サポート

F95 の文 `USE SUNPERG` をプログラムに含めることにより、Performance Library のモジュールと定義を利用することができます。これにより、以下の機能が得られます。

- 型への非依存性。FORTRAN 77 のルーチンでは、名前の一部として型を指定する必要がありました。Fortran 95では、特定のデータ型のルーチンは、そのルーチンに引き渡した引数のデータ型によって決めることができます。
- コンパイル時チェック。FORTRAN 77 では、一般的には、どのようなパラメータを特定のルーチンに引き渡すべきかをコンパイラが知ることはできません。Fortran 95 では、`USE SUNPERF` 文を使用することにより、コンパイラは各 Sun Performance Library ルーチンに引き渡される各パラメータの数、型、サイズ、および形状がどのようなものであるべきかを知ることができます。コンパイラは、呼び出しの内容をその期待値と照合し、コンパイル中にエラーを見つけ出します。
- 省略可能なパラメータ。FORTRAN 77 では、すべてのパラメータをすべてのルーチンについて順に指定する必要があります。Fortran 95 では、一部のパラメータが省略可能となっています。Sun Performance Library では、すべての増分パラメータ (`INCX`、`INCY` など)、ワークスペース、先頭次元 (`LDA`、`LDB` など)、長さ/サイズパラメータが省略可能です。

これらの機能の使い方や例については、『Sun Performance Library ユーザーズガイド』を参照してください。

Sun Performance Library のライセンス供与に関する変更

Sun Performance Library は、もはやライセンスを必要としません。しかし、`-lsunperf` ではなく、引き続き `-xlic_lib=sunperf` を使用してリンクする必要があります。`-xlic_lib=sunperf` を使用して、次のことを確実に行ってください。

- アプリケーションを正しいサポートライブラリにリンクします。このリリースでは、Sun Performance Library は Fortran 77 ではなく Fortran 95 でコンパイルされています。`-xlic_lib` を使うと、Fortran 77 の実行時ライブラリではなく Fortran 95 の実行時ライブラリにリンクします。
- 正しいバージョンの Sun Performance Library を使用します。`-subparallel` を使って構築したプログラムと `-subparallel` を使わないで構築したプログラム、および異なる `-xarch` の値では、サポートする Sun Performance Library のバージョンが異なります。`-xlic_lib=sunperf` を指定すると、ドライバは、使用中のコマンド行オプションに最適な Sun Performance Library のバージョンを使用します。

標本アナライザ

Sun WorkShop 6 標本アナライザは、Sun WorkShop 5.0 に装備されたアナライザを全面的に書き換えたものです。

表 3-12 は、Sun WorkShop 6 標本アナライザの新機能です。一部の製品の機能については、表の次の節でさらに詳しく説明しています。

表 3-12 標本アナライザの新機能

機能	説明
関数分析	
主ディスプレイ (関数リスト)	新リリースにおける主ディスプレイは「関数リスト」であり、アナライザ起動時にデフォルトで表示されます。
複数の測定値	新リリースの「関数リスト」では同時に複数の種類の測定値が表示されるので、一度に表示させたい測定値 1 つを選択する必要はありません。「関数リスト」は測定値を絶対値または百分率で表示します。
「概要メトリック」ウィンドウ	「表示」メニューから利用できる新しい「概要メトリック」ウィンドウでは、選択されている関数について記録されている測定値を、値と百分率の両方で表示します。「概要メトリック」ウィンドウの内容は、関数リスト表示に表示される内容と無関係です。
「呼び出し元 - 呼び出し先」ウィンドウ	「関数リスト」から新機能である「呼び出し元 - 呼び出し先」ウィンドウにアクセスできます。このウィンドウには、関数の呼び出し側からその関数の被呼び出し側に測定値がどのように帰属されるのかが示されます。
注釈付きソースコードの生成	選択した関数について注釈付きソースコードを生成し、その結果を編集ウィンドウに表示できるようになりました。
注釈付き逆アセンブリの生成	選択した関数について注釈付き逆アセンブリを生成し、その結果を編集ウィンドウに表示できるようになりました。

表 3-12 標本アナライザの新機能 (続き)

機能	説明
サンプル、スレッド、LWP によるデータのフィルタリング	新リリースでは、サンプル、スレッド、または LWP あるいはこれらを組み合わせたものによるデータのフィルタリングを「フィルタの選択」ダイアログボックスで行えます。この結果、選択されたサブセットから取り出されたデータだけを示すように、あらゆるディスプレイとウィンドウが更新されます。
スレッド同期待機測定値	新リリースでは、指定のしきい値を超える同期イベント数とこれらのイベントからの総遅延との 2 種類のスレッド同期待機測定値を利用できます。「測定値」の詳細は、98 ページの「測定値」を参照してください。
複数の実験のロード	新リリースでは、複数の実験を一度にアナライザにロードできます。これらの実験の測定値が「関数リスト」ディスプレイに表示されます。

主ディスプレイ (関数リスト)

新リリースでは、「関数リスト」が主ディスプレイであり、アナライザ起動時にデフォルトで表示されます。

関数リストでは同時に複数の種類の測定値が表示されるので、ディスプレイの型だけを変更するように「データ」リストボックスが変更されました。「ディスプレイリスト」オプションメニューは削除されました。

このリリースでは、「関数リスト」に記載されている測定値を秒/カウントによる絶対値またはプログラム全体の値に対する百分率、またはその両方の形で確認できます。「関数リスト」ディスプレイから「メトリックの選択」ダイアログボックスを呼び出して、以下が実行できます。

- 「関数リスト」に表示された測定値の選択
- 測定値をカウント、百分率のどちらで表示するか、あるいは両方で表示するか
- どの測定値で「関数リスト」をソートするか
- リストの並べ替え

「呼び出し元 - 呼び出し先」ウィンドウ

「関数リスト」から新機能である「呼び出し元 - 呼び出し先」ウィンドウにアクセスできます。このウィンドウには、関数の呼び出し元からその関数の呼び出し先に測定値がどのように帰属されるのかが示されます。「呼び出し元 - 呼び出し先」ウィンドウでは、選択されている関数がディスプレイ中央に表示され、この関数の呼び出し側は上のパネルに、この関数の呼び出し元は下のパネルに示されます。選択された関数について、この関数内部における使用率が、帰属測定値によって示されます。上の呼び出し元については、選択されている関数内とこの関数が呼び出す全関数内における使用率が、呼び出しスタックを呼び出し元までたどって示されます。下の呼び出し元については、選択されている関数からの呼び出しに帰属する呼び出し元測定値の割合が示されます。

呼び出し元パネルと呼び出し先パネルのどちらかで関数をクリックすれば、「呼び出し元 - 呼び出し先」ウィンドウでプログラムの構造内をナビゲートできます。ディスプレイは、新たに選択された関数を基準としてセンタリングし直します。

注釈付きソースコードの生成

新リリースでは、選択した関数について注釈付きソースコードを生成し、その結果を編集ウィンドウに表示できるようになりました。ソースコードには、「関数リスト」と同じ測定値セットを使った、行単位の測定値の注釈が付きます。コンパイラの並列化のコメントと Fortran 95 の copyin/copyout のコメントも、ソースとインタリーブされた状態で入ります。ソースコードディスプレイでは、`-g` によるコンパイルを必要とし、最適化コードに対して有効となります (`-g` では最適化と並列化を使用不可にすることはできなくなりました)。

注釈付き逆アセンブリの生成

選択した関数について注釈付き逆アセンブリを生成し、その結果を編集ウィンドウに表示できるようになりました。逆アセンブリには、「関数リスト」と同じセットを使った、命令単位の測定値の注釈が付きます。また逆アセンブリには、コンパイラのコメントとインタリーブされたソースも入ります。

測定値

Sun WorkShop 6 アナライザでは、以下のとおり、新しい測定値が加わり、既存の測定値が変更されました。

- 実行プロファイルデータは、新リリースでは時間ベースのプロファイリングと呼びます。実行プロファイルデータには常に呼び出された関数の回数が含まれます。
- ヒストグラムデータは、新リリースでは排他的測定値と呼びます。
- 累積データ測定値は、新リリースでは包含的測定値と呼びます。
- クロックベースのプロファイルデータは以下の測定値を生成します。
  - LWP 総時間
  - ユーザー CPU 時間
  - システム CPU 時間
  - システム待ち時間
  - テキストページフォルト時間
  - データページフォルト時間
  - 時計時間
- 新しい種類のデータであるスレッド同期待機遅延追跡が装備され、以下の測定値を生成します。
  - 指定のしきい値を超える同期イベント数
  - これらのイベントからの総遅延

  これらの測定値はデフォルトでは集められません。

その他の変更事項

Sun WorkShop 6 標本アナライザでは、その他にも以下の変更を加えています。

- 「概要」ディスプレイでサンプルをクリックしてサブセットを選択する方法は、このリリースでは使用できなくなっています。「標本の選択」テキストボックスと矢印ボタンは新リリースのアナライザには備わっていないので、「表示」メニューに選択コマンドが表示されることはありません。サンプル、スレッド、LWP のフィルタ選択は、すべて「フィルタの選択」ダイアログボックスで行なってください。
- アナライザの中から実験記録ファイルを削除できなくなっています。「実験」➤「削除」 コマンドはなくなり、代わりにアナライザから実験を解除できる 「実験」➤「解除」 コマンドが用意されています。`er_rm` コマンドによって削除しないかぎり、実験記録ファイルはディスク上に存在したままとなります。
- `er_export` ASCII データのフォーマットが完全に変更されました (`er_export` はデバッグをする場合に限り使用されます)。

インストール

表 3-13 は、Sun WorkShop 6 の新しいインストール機能です。

表 3-13 インストールの新機能

機能	説明
Web Start	Web Start は、新しい GUI インストールソフトウェアです。
FLEXlm 7.0	FLEXlm 7.0 は、Sun WorkShop 6 とともに提供されるライセンスマネージャソフトウェアです。

HTML 形式のドキュメント

Sun WorkShop 6 および Sun WorkShop TeamWare 6 のマニュアル、マニュアルページと README は、テキスト形式のファイルとともに HTML 形式のファイルも用意されています。オンラインヘルプはこのバージョンから HTML 形式に変わっています。

HTML 形式で利用できる Sun WorkShop 6 のマニュアルを表示するには、Netscape™ Communicator 4.0 または同互換の Netscape バージョンが必要です。Netscape™ Communicator は Solaris™ 7 オペレーティング環境 と Solaris™ 8 オペレーティング環境に含まれています。

Solaris 2.6 オペレーティング環境を使用していて Netscape Communicator 4.0 または同互換バージョンがお手元にない場合は、以下の Netscape Communications Corporation の Web サイトから Netscape Communicator 4.7 をダウンロードできます。

```
http://www.netscape.com/download/index.html
```

Sun WorkShop のオンラインヘルプ (HTML 形式) では、JavaScript が使用可能になっていなければなりません。JavaScript は Netscape のデフォルトの設定では、使用可能になっています。

HTML 形式のインストール済み Sun WorkShop 6 ドキュメントにアクセスするには、ブラウザで次のファイルを指定してください。

```
file:/opt/SUNWspro/docs/ja/index.html
```

Sun WorkShop 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。