



Sun WorkShop の概要

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. 816-0884-01
2001 年 8 月 Revision A

本製品およびそれに関連する文書は、その使用、複製、頒布および逆コンパイルを制限するライセンスのもとにおいて頒布されます。サン・マイクロシステムズ株式会社の書面による事前の許可なく、本製品および関連する文書のいかなる部分も、いかなる方法によっても複製することが禁じられます。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。Netscape™、Netscape Navigator™、および Netscape Communications Corporation のロゴは、次の著作権で保護されています。
© 1995 Netscape Communications Corporation.

Sun、Sun Microsystems、docs.sun.com、AnswerBook2、SunOS、JavaScript、SunExpress、Sun WorkShop、Sun WorkShop Professional、Sun Performance Library、Sun Performance WorkShop、Sun Visual WorkShop、Forte 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 商標は、米国 SPARC International, Inc. のライセンスを受けて使用している同社の米国およびその他の国における商標または登録商標です。SPARC 商標が付いた製品は、米国 Sun Microsystems 社が開発したアーキテクチャに基づくものです。

本書で参照されている製品やサービスに関しては、該当する会社または組織に直接お問い合わせください。

OPEN LOOK および Sun Graphical User Interface は、米国 Sun Microsystems 社が自社のユーザおよびライセンス実施権者向けに開発しました。米国 Sun Microsystems 社は、コンピュータ産業用のビジュアルまたはグラフィカル・ユーザインタフェースの概念の研究開発における米国 Xerox 社の先駆者としての成果を認めるものです。米国 Sun Microsystems 社は米国 Xerox 社から Xerox Graphical User Interface の非独占的ライセンスを取得しており、このライセンスは米国 Sun Microsystems 社のライセンス実施権者にも適用されます。

Sun f90 / f95 は、米国 Cray Inc. の Cray CF90™ に基づいています。

Federal Acquisitions: Commercial Software -- Government Useres Subject to Standard License Terms and Conditions.

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。

本製品が、外国為替および外国貿易管理法(外為法)に定められる戦略物資等(貨物または役務)に該当する場合、本製品を輸出または日本国外へ持ち出す際には、サン・マイクロシステムズ株式会社の事前の書面による承諾を得ることのほか、外為法および関連法規に基づく輸出手続き、また場合によっては、米国商務省または米国所轄官庁の許可を得ることが必要です。

原典：	<i>Introduction to Sun WorkShop</i> Part No: 806-7980-10 Revision A
-----	---

© 2001 by Sun Microsystems, Inc.



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

製品名の変更について iii

はじめに xvii

1. Sun WorkShop 統合プログラミング環境について 1

統合テキストエディタ 1

 XEmacs の Mule 機能 2

コンパイラ 3

統合デバッグ 4

ソースコードブラウザ 5

パフォーマンス、ソースコード管理、GUI 構築ツール 5

 マルチスレッド開発ツール 6

 Sun WorkShop TeamWare 6

 Sun WorkShop Visual 6

2. Sun WorkShop の使用 7

プロジェクトの操作 7

 プロジェクトの作成 9

 プロジェクトターゲットの構築 10

 プロジェクトの編集 11

メインウィンドウの使用方法	11
テキストエディタの選択とテキストエディタのオプション	11
起動オプションとプロジェクトオプションの設定	12
Sun WorkShop ツールの使用方法	13

3. プログラムの構築 15

ターゲットの操作	15
Sun WorkShop ターゲット	15
ユーザーメイクファイルターゲット	16
「構築」ウィンドウ	17
プログラムの構築	18
デフォルト値を使用した構築	19
独自の構築値の指定	20
構築オプションの指定	20
メイクファイルマクロの使用方法	21
環境変数の使用	21
構築エラーの表示	21
構築の終了	23

4. プログラムのデバッグ 25

デバッグの準備	26
デバッグの開始	26
デバッグウィンドウのカスタマイズ	29
コードのステップ実行	29
ブレークポイントの設定	30
値とデータの検査	31
データ値の監視	31
パフォーマンスデータの収集	32

実行時エラーの検出	32
コードのトレース	33
呼び出しスタックの検査	34
マルチスレッドプログラムのデバッグ	35
複数プロセスの同時デバッグ	35
セッションの管理	35
子プロセスのデバッグ	36
デバッグの終了	37
5. ソースコードのブラウズ	39
パターン検索モード	39
パターン検索で使用できる特殊文字	41
複数ディレクトリの検索	41
ソースブラウズモードの使用方法	42
ソースブラウズ用データベース	43
ソースブラウズで使用できる特殊文字	43
複数ディレクトリのブラウズ	44
ブラウズとグラフ表示の関係	45
関数のグラフ表示	45
クラスのグラフ表示	47
クラスのブラウズ	48
ブラウズの終了	51
6. プログラムのパフォーマンスの解析	53
パフォーマンスデータの収集	54
パフォーマンスデータの解析	55
関数およびロードオブジェクトデータの表示	56
呼び出し元と呼び出し先のデータの表示	56

注釈付きのソースコードと逆アセンブリコードの表示 56

7. ソースファイルのマージ 57

ファイルの読み込み 57

相違の処理 60

 ファイルマージのグリフの意味 60

 相違間の移動 61

 相違の解決 62

 相違オプションの設定 62

ファイルの自動マージ 62

出力ファイルの保存 63

ファイルマージオプションの設定 63

A. Sun WorkShop とテキストエディタのリソース 65

リソース設定の変更 65

編集可能な Sun WorkShop のリソース 67

 エディタウィンドウの強調色 68

 「データグラフ」ウィンドウの色 68

 「コールグラフ」ウィンドウと「クラスグラフ」ウィンドウの色 69

警告音 69

デバッグボタン 70

 「dbx コマンド」ウィンドウと「プログラム入出力」ウィンドウの出力行 70

プロジェクト構築コマンド 71

 「Web での最新情報」の表示に使用するブラウザ 71

ハイパーリンクウィンドウの文字フォント 71

ハイパーリンク用リソース 73

テキストの自動改行 74

垂直スクロールバー 74

Motif に固有のリソース	74
ウィンドウの前景色と背景色	76
スクロールバーの背景とトグルボタンの色	77
編集可能なテキストエディタのリソース	77
テキストエディタのデフォルトパス名	78
点滅ポインタ	78
Motif 環境のテキストエディタのフォント	79
テキストエディタウィンドウの色	79
スクロールリストの背景色	79
書き込み可能領域の背景色	80
式のバルーン評価ポップアップウィンドウの寸法	80
警告音	80
B. make ユーティリティとメイクファイル	81
メイクファイル	82
FORTRAN 77 の例	82
C++ の例	83
make ユーティリティ	84
マクロ	85
C. dmake ユーティリティの使用	89
基本概念	89
dmake ホスト	90
構築サーバー	93
dmake ユーティリティとメイクファイル	94
ターゲットの並列構築	95
メイクファイルに対する制限事項	95
並列処理に対する制限	99

D. sbquery、sb_init、sbtags によるソースのブラウズ 101

sbquery ユーティリティ 101

sbquery のオプション 102

環境変数 105

sb_init ファイルと sb_init コマンド 106

sbtags ユーティリティ 110

用語集 113

索引 117

図目次

- 図 2-1 メインウィンドウ 11
- 図 3-1 構築ウィンドウ 18
- 図 3-2 「ターゲットの新規定義」ダイアログボックス 19
- 図 3-3 「構築」ウィンドウの構築出力表示区画に表示された構築エラーメッセージ 22
- 図 3-4 構築エラーと、エラーに関する情報を示すダイアログボックス 23
- 図 4-1 Sun Workshop の「デバッグ」ウィンドウ 28
- 図 5-1 パターン検索モードでの「ブラウズ」ウィンドウ 40
- 図 5-2 ソースブラウズモードのときの「ブラウズ」ウィンドウ 42
- 図 5-3 ブラウズ、クラスグラフ、クラスブラウザの相互関係 45
- 図 5-4 「コールグラフ」ウィンドウ 47
- 図 5-5 「クラスグラフ」ウィンドウ 48
- 図 5-6 「クラスブラウザ」ウィンドウ 50
- 図 7-1 「ファイルマージ」ウィンドウ 59

表目次

表 5-1	パターン検索で使用できる特殊文字	41
表 5-2	ソースブラウザで使用できる特殊文字	44
表 6-1	収集できるデータの種類	54
表 6-2	表示、解析できるデータの種類	55
表 A-1	エディタウィンドウで使用される強調色のリソース	68
表 A-2	「データグラフ」ウィンドウの色のリソース	68
表 A-3	「コールグラフ」と「クラスグラフ」のウィンドウで使用される色のリソース	69
表 A-4	警告音の有効 / 無効を切り替えるリソース	69
表 A-5	デバッグのボタンが無効になるまでの遅延時間を指定するリソース	70
表 A-6	「dbx コマンド」ウィンドウおよび「プログラム入出力」ウィンドウに保存される出力行数を設定するリソース	70
表 A-7	プロジェクト構築コマンドのリソース	71
表 A-8	「Web での最新情報」の表示に使用するブラウザのパスを変更するリソース	71
表 A-9	ハイパーリンクフォントのリソース	73
表 A-10	複数バイト文字用のハイパーリンクフォントのリソース	73
表 A-11	テキストの自動改行を設定するリソース	74
表 A-12	垂直スクロールバーの有効 / 無効を切り替えるリソース	74
表 A-13	Motif (非 CDE) 固有のフォントのリソース	74
表 A-14	個々のウィンドウのフォントのリソース	75
表 A-15	テーブル形式のウィンドウで使用されるフォントのリソース	75
表 A-16	Sun WorkShop のウィンドウ、ダイアログボックス、メニュー、ボタンで使用される色のリソース	76

表 A-17	スクロールバーの背景やトグルボタンの色のリソース	77
表 A-18	テキストエディタのデフォルトパスを設定するリソース	78
表 A-19	点滅ポインタのリソース	78
表 A-20	Motif (非 CDE) 固有のフォントのリソース	79
表 A-21	テキストエディタのウィンドウ、ダイアログボックス、メニュー、ボタンに使用する色のリソース	79
表 A-22	スクロールリストの背景色を設定するリソース	79
表 A-23	書き込み可能なテキスト領域の色を設定するリソース	80
表 A-24	バルーン評価ポップアップウィンドウのサイズ	80
表 A-25	警告音のオン/オフを切り替えるリソース	80
表 D-1	sbquery のオプション	102
表 D-2	言語フィルタオプション	104
表 D-3	フォーカスオプション	105
表 D-4	環境変数	105
表 D-5	sb_init で使用できるコマンド	106

コード例目次

コード例 B-1	Fortran 77 用のメイクファイル例	83
コード例 B-2	C++ 用のメイクファイル例	84
コード例 B-3	make のデフォルトの接尾辞規則	85
コード例 C-1	.dmake.rc ファイル	90
コード例 C-2	構築サーバーグループを定義した .dmake.rc ファイル	91
コード例 C-3	構築サーバーの代替パスを指定した .dmake.rc ファイル	92
コード例 C-4	特殊文字を使用した .dmake.rc	92
コード例 C-5	dmake.conf ファイル	93

はじめに

このマニュアルでは、Sun WorkShop™ 統合プログラミング環境の基本的なプログラム開発機能について説明します。このマニュアルは Fortran、C、C++、Solaris™ オペレーティング環境および UNIX® コマンドに関する実用的な知識を持つアプリケーション開発者を対象にしています。

Web から Sun WorkShop ツールの最新情報を入手するには、「Web での最新情報」ダイアログボックスを使用してください。「Web での最新情報」ダイアログボックスを開くには、Sun WorkShop のウィンドウで「ヘルプ」▶「Web での最新情報」を選択します。

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。	<pre>machine_name% su Password:</pre>
AaBbCc123 または ゴシック	コマンド行の可変部分。実際の名前または実際の値と置き換えてください。	rm <i>filename</i> と入力します。 rm ファイル名 と入力します。
『』	参照する書名を示します。	『SPARCstorage Array ユーザーマニュアル』
「」	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、次の指示に従って 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 統合プログラミング環境について

Sun WorkShop は、C++、C、Fortran 77/95 ソフトウェア開発プロジェクトの構築、編集、デバッグ、ソースブラウザ、チューニングのための統合ツールを提供することによって、複雑な開発作業を簡略化する統合プログラミング環境です。

Sun WorkShop は、次のコンポーネントで構成されています。

- 統合テキストエディタ
- コンパイラ
- 統合デバッグ
- ソースコードブラウザ
- パフォーマンス、ソースコード管理、GUI 構築ツール

注 - この章に記述されているマニュアルにアクセスするには、xxi 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。

統合テキストエディタ

テキストエディタは、Sun WorkShop 統合プログラミング環境の中核ツールです。Sun WorkShop 統合プログラミング環境では、テキストエディタから式の評価、ブレークポイントの設定、関数のステップ実行を行うことができます。

Sun WorkShop 6 update 2 には、次の統合エディタが用意されています。

- **NEdit**: X/Motif システム用のグラフィカルユーザーインターフェイス形式のプレーンテキストエディタ。デフォルトでは、Sun WorkShop は、このエディタを使用します (テキストエディタの変更については、11 ページの「テキストエディタの選択とテキストエディタのオプション」を参照)。NEdit については、オンラインヘルプと、NEdit の Web ページ (<http://www.nedit.org>) を参照してください。
- **XEmacs**: アプリケーション開発システムとしての機能も備えた、カスタマイズ可能なテキストエディタ。XEmacs についての詳細は、オンラインヘルプと、XEmacs の Web ページ (<http://www.xemacs.org>) を参照してください。
- **GNU Emacs**: マニュアル類が充実していて、拡張性が高く、カスタマイズ可能なリアルタイムディスプレイエディタ。GNU Emacs についての詳細は、オンラインヘルプと、GNU の Web ページ (<http://www.gnu.org>) を参照してください。
- **vi**: UNIX システム標準のスクリーンエディタ。vi についての詳細は、オンラインヘルプの該当項目を参照してください。
- **vim**: UNIX システム標準の vi テキストエディタ (グラフィカルユーザーインターフェイスオプション付き) の改良版。vim についての詳細は、オンラインヘルプと、vim の Web ページ (<http://www.vim.org>) を参照してください。

注 – NEdit と vim は日本語化されていないため、日本語環境で日本語を扱うことはできません。また、純正の GNU Emacs も日本語を扱うことはできません。GNU Emacs の多国語対応版である Mule は日本語を扱うことができます。日本語環境で WorkShop を起動した場合に、Mule をエディタとして起動するように設定できます。

XEmacs の Mule 機能

XEmacs には Mule 機能が含まれているため、日本語を扱うことができます。

WorkShop のエディタのオプションで選択可能なエディタと起動されるコマンドは、以下の表のように環境によって異なります。

英語環境		日本語環境	
選択可能エディタ	起動するコマンド	選択可能エディタ	起動するコマンド
NEdit	付属の nedit	Vi	付属の Vi エディタ
Vi	付属の Vi エディタ	Mule	PATH 中の mule
GNU Emacs	PATH 中の emacs	XEmacs	付属の xemacs-mule
XEmacs	付属の xemacs		
Vim	付属の gvim		

Sun WorkShop のエディタについては、次を参照してください。

- 11 ページの「テキストエディタの選択とテキストエディタのオプション」
- オンラインヘルプ

コンパイラ

Sun WorkShop 6 update 2 では、次のコンパイラをサポートしています。

- C++ コンパイラ

この C++ コンパイラ (cc) は、C++ に関する ISO 規格の ISO IS 14882:1998, Programming Language C++ に対応しています。ただし、この規格にある「テンプレートはテンプレートパラメータとユニバーサル文字名である」という仕様には対応していません。C++ コンパイラについては、『C++ ユーザーズガイド』を参照してください。

- C コンパイラ

このコンパイラは、1990 ISO (ANSI) C 言語、環境規格およびあらゆる改訂に完全準拠しているとともに、従来の K&R C にも対応しています。また、OpenMP C/C++ アプリケーションプログラムインタフェースバージョン 1.0 仕様にも対応しています。C オプティマイザによって、最適化されていないコードよりも大幅にパフォーマンスが向上します。このコードオプティマイザは、冗長なコードを取り除いて、レジスタを効率よく割り当て、命令をスケジュールします。また、インク

リメンタルリンカーという、デバッグ段階でのリンク時間を短縮する機能も用意されています。C コンパイラについては、『C ユーザーズガイド』を参照してください。

- Fortran コンパイラ

- Fortran 95

Fortran 95 コンパイラには、Fortran 95 ISO/IEC 1539:1997 規格で定義されている仕様が完全に実装されています。この規格では、多数の機能が追加されています。Fortran 95 コンパイラは、OpenMP 2.0 マルチプログラミングインタフェースも実装しています。また、Fortran 95 コンパイラは、Sun WorkShop の他のコンポーネントと連携して自動的にコードを並列化します。Fortran 95 コンパイラについては、『Fortran ユーザーズガイド』または『Fortran プログラミングガイド』を参照してください。

- Fortran 77

Fortran 77 コンパイラには、Fortran 77 ANSI X3.9-1978 および ISO 1539-1980 規格で定義されている仕様が完全に実装されています。また、VAX VMS Fortran および Cray Fortran 互換を実現する機能が追加されています。Fortran 77 コンパイラは、Sun WorkShop の他のコンポーネントと連携して自動的にコードを並列化します。Fortran 77 コンパイラについては、『Fortran ユーザーズガイド』または『Fortran プログラミングガイド』を参照してください。

統合デバッグ

Sun WorkShop 統合プログラミング環境では、ソースコードデバッグウィンドウを使用して、プログラムの実行を制御し、停止したプログラムの状態を検査する機能があります。大部分のデバッグ操作は、「デバッグ」ウィンドウとその下位のデバッグウィンドウから実行できます。また、基本的なデバッグ操作は、デバッグ目的でプログラムを読み込んだときに自動的に開くテキストエディタのウィンドウから実行することもできます。Sun WorkShop では、プログラムを動的に実行して完全に制御し、パフォーマンスデータを収集できます。また、dbx という、コマンド行で行うソースコードデバッグも付属しています。

統合デバッグサービスについては、次を参照してください。

- このマニュアルの第 4 章
- オンラインヘルプ

ソースコードブラウズ

Sun WorkShop では、パターン検索モードまたはソースブラウズモードで「ブラウズ」ウィンドウから問い合わせを発行することによって、C、C++、Fortran 77/95 で書かれたソースコードをブラウズできます。パターン検索モードでは、ソースコードから、コメント内のテキストなどの任意の文字列を検索できます。ソースブラウズモードでは、ソースブラウズオプションでソースファイルをコンパイルしたときに作成されるデータベースに基づいて、プログラムに定義されている特定のシンボルに一致するものすべてを見つけることができます。プロジェクトウィザードでプロジェクトを作成または編集する際にコードをコンパイルするときにデータベースを作成することを選択できます。問い合わせ内容に一致する文字列またはシンボルは、「ブラウズ」ウィンドウの一致区画に前後のソースコードと一緒に表示されます。

プログラム内の関数とサブルーチンの関係をグラフで表示することもできます。また、ソースコードが C++ で書かれている場合は、プログラムに定義されているクラスをブラウズしたり、グラフで表示したりできます。

ソースコードブラウザについては、次を参照してください。

- このマニュアルの第 5 章
- オンラインヘルプ

パフォーマンス、ソースコード管理、GUI 構築ツール

Sun WorkShop のメインウィンドウには、「ツール」メニューがあり、デフォルトでは、このメニューから「パフォーマンスアナライザ」（プログラムのパフォーマンス解析）や「ファイルマージ」（Sun WorkShop TeamWare ソースコード管理ツールの 1 つ）にアクセスできます。この他、C++ コンパイラを使用している場合は、GUI 構築ツールの Sun WorkShop Visual にアクセスできます。

マルチスレッド開発ツール

Sun WorkShop 統合プログラミング環境には、マルチスレッドアプリケーションを開発するためのツールが付属しています。Sun WorkShop のデバッグウィンドウを使用して、マルチスレッドプログラムを動的に解析、制御できます。ロック lint は、ソースコードを解析して、デッドロックやデータ競合状態などの同期エラーがないかどうかを調査します。標本コレクタおよびパフォーマンスアナライザを使用して、マルチスレッドプログラムの関数、ロードオブジェクト、標本間隔、またはマルチスレッドプログラムのスレッドや軽量プロセス(LWPs)によって分類される広範囲のメトリック(計測データ)を調査できます。

マルチスレッド開発ツールについては、次を参照してください。

- このマニュアルの第 6 章
- 『プログラムのパフォーマンス解析』
- オンラインヘルプ

Sun WorkShop TeamWare

Sun WorkShop TeamWare ソースコード管理ツールによって、GUI またはコマンド行のどちらからでもソースコードファイルを管理できます。チームのメンバーが複数サイトに分散している場合でも、製品の調整、統合、構築を並行して行うことができます。

TeamWare については、次を参照してください。

- このマニュアルの第 7 章
- 『Sun Workshop TeamWare ユーザーズガイド』
- オンラインヘルプ

Sun WorkShop Visual

Sun WorkShop Visual は、グラフィカルユーザーインターフェース (GUI) の設計や、移植可能なオブジェクト指向型のコードの作成、Motif や Java、Microsoft Foundation Class 用 GUI の開発を支援します。GUI のコードは、設計が完了する自動的に作成されます。

Visual については、『Sun WorkShop Visual ユーザーズガイド』を参照してください。

第2章

Sun WorkShop の使用

Sun WorkShop をインストールして、アクセスが可能になると (xix ページの「Sun WorkShop の開発ツールとマニュアルページへのアクセス」を参照)、コマンド行から次のように入力することによって、Sun WorkShop 統合プログラミング環境を起動できます。

```
% workshop&
```

workshop コマンドについては、workshop(1) のマニュアルページを参照してください。

この章では、Sun WorkShop で作業を開始する方法と、Sun WorkShop 統合プログラミング環境における次の基本事項を説明します。

- プロジェクトの操作
- メインウィンドウの使用法

これらに関する具体的な手順とその他の情報については、Sun WorkShop のオンラインヘルプを参照してください (オンラインヘルプには、Sun WorkShop のどのウィンドウからでも「ヘルプ」メニューまたは「ヘルプ」ボタンでアクセスできます)。

プロジェクトの操作

Sun WorkShop 6 update 2 では、開発プロジェクトに関係するすべてのファイル、プログラム、ターゲットを記録したプロジェクトを使用して、メイクファイルを記述することなくプログラムを構築します。プロジェクトは、実行可能ファイル、静的ライブラリやアーカイブ、共有ライブラリ、Fortran アプリケーション、複合アプリケー

ション、ユーザーメイクファイルアプリケーションを構築するとき使用するファイルやコンパイラオプション、デバッグオプション、構築関連のオプションをまとめたリストです。

Sun WorkShop 統合プログラミング環境を起動すると、「Sun WorkShop へようこそ」ダイアログボックスが開き、Sun WorkShop プロジェクトおよびプロジェクトウィザードにアクセスできます。ここで、説明区画に表示される「プロジェクト」のリンクをクリックすると、オンラインヘルプのプロジェクトに関する情報が表示されます。「Sun WorkShop へようこそ」ダイアログボックスにある「ヘルプ」をクリックして、このダイアログボックスに関する情報を表示することもできます。

「Sun WorkShop へようこそ」ダイアログボックスまたは Sun WorkShop のメインウィンドウの「プロジェクト」メニューにあるコマンドを使用して、次の作業を行うことができます。

- プロジェクトウィザードとユーザー自身が作成したメイクファイル、または Sun WorkShop が作成したメイクファイルを使用した新規プロジェクトの作成や、簡単なプログラムの構築。詳細はオンラインヘルプを参照。
- プロジェクトのコンパイル方法やソースブラウザ用の情報作成などの、既存プロジェクトの設定の変更。詳細はオンラインヘルプを参照。

Sun WorkShop ワークセットが存在する場合は、そのワークセットを読み込むことによって、自動的にプロジェクトに変換できます。具体的な手順については、オンラインヘルプを参照してください。

また、プロジェクトを読み込まずに Sun WorkShop 統合プログラミング環境を利用することもできます。ピックリストは、開発プロジェクトに関するファイルやプログラム、ディレクトリの記録です (ピックリストについては、オンラインヘルプを参照)。このピックリストを使用し、Sun WorkShop のメインウィンドウのメニューから、各ファイルにアクセスして、ターゲットを構築し、実行可能ファイルをデバッグできます。

プロジェクトの作成

「Sun WorkShop へようこそ」ダイアログボックスまたはメインウィンドウの「プロジェクト」メニューからプロジェクトウィザードを起動して、プロジェクトを作成できます。「Sun WorkShop へようこそ」ダイアログボックスからは、次のことを行うことができます。

- | | |
|------------|--|
| 新規プロジェクト作成 | 「新規プロジェクト作成」ウィザードからの指示に従って操作していくことによって、既存のソースファイルから新規プロジェクトを作成できます。プロジェクトの種類を選択すると、その種類の新規プロジェクトを作成するために必要な情報の入力が求められます。 |
| 単純なプログラム構築 | 「単純なプログラム構築」ウィザードからの指示に従って操作していくことによって、一群のソースファイルから簡単な実行可能ファイルを構築できます。使用できるコンパイルオプションは限られており、標準のライブラリしかリンクできません。 |
| 空のプロジェクト作成 | 「空のプロジェクト作成」ダイアログボックスが開き、既存のソースファイルなしで新規プロジェクトを作成するよう指示します。テキストエディタのウィンドウが開いて、プログラムの作成を開始できます。 |

「新規プロジェクト作成」ウィザードでは、プロジェクトを定義するよう求められます。Sun WorkShop 統合プログラミング環境は、ここで定義されたタイプの、指定されたソースファイル情報を含むプロジェクト (ファイル名の拡張子が .prd のファイル) を作成します。

このプロジェクトファイルの情報は、プロジェクトファイルへの絶対パス (フルパス) と相対的なプロジェクトディレクトリ (ファイル名かベース名のみ) 情報の入ったプロジェクトを作成することによって、開発チームの他のメンバーと共有できます。たとえば、プロジェクト用のワークスペースがあり、そのワークスペースの最上位のディレクトリに `ws.prd` というプロジェクトを作成した場合、プロジェクトファイル名とプロジェクトディレクトリは、次のようになります。

プロジェクトファイル名: `/home/workspaces/ws/ws.prd`
プロジェクトディレクトリ: `.`

開発チームのメンバーは、`/export/myws` にこのワークスペースのコピーを保持できます。この後、コピーを保持するメンバーが `/export/myws/ws.prd` プロジェクトファイルを開くと、そのメンバーのローカルファイルにだけプロジェクトが適用されます。その他のメンバーのファイルにはプロジェクトは適用されません。

プロジェクトターゲットの構築

プロジェクトを作成すると、次のいずれかの操作を行うことによってプロジェクトターゲットを構築できます。

- メインウィンドウのツールバーから「構築」ボタンをクリックする。
- メインウィンドウから「構築」▶「プロジェクト構築」を選択する。
- コマンド行から `makeprd` コマンドを使用する (たとえば、この機能を使用し、スクリプトまたは `cron` ファイル内からプロジェクトを構築可能)。

これらのいずれかの操作が行われると、Sun WorkShop 統合プログラミング環境は次のことを行います。

1. 「新規プロジェクト作成」ウィザードまたは「現在のプロジェクトを編集」ウィンドウで定義されたプロジェクトの情報に基づいてメークファイルを作成します。
2. 「構築」ウィンドウを開きます。
3. `make` ユーティリティを起動します。
4. 「構築」ウィンドウに構築結果を表示します。

「構築」ウィンドウ内で構築エラーのハイパーテキストリンクをクリックすると、テキストエディタが開いて、構築エラーの部分が強調表示されるため、エラーを調べて、修正できます。

プロジェクトターゲットの構築については、次を参照してください。

- このマニュアルの第 3 章
- オンラインヘルプ
- `makeprd(1)` のマニュアルページ

プロジェクトの編集

「現在のプロジェクトを編集」ウィンドウを使用し、プロジェクトを編集できます。「現在のプロジェクトを編集」ウィンドウを開くには、Sun WorkShop のメインウィンドウから「プロジェクト」▶「プロジェクトの編集」を選択します。詳細は、オンラインヘルプを参照してください。

メインウィンドウの使用方法

メインウィンドウから、アプリケーションを作成、開発、デバッグ、チューニングするためのツールにアクセスできます。また、このウィンドウからは、デフォルトのテキストエディタを選択したり、さまざまな種類のオプションを設定したりすることもできます。

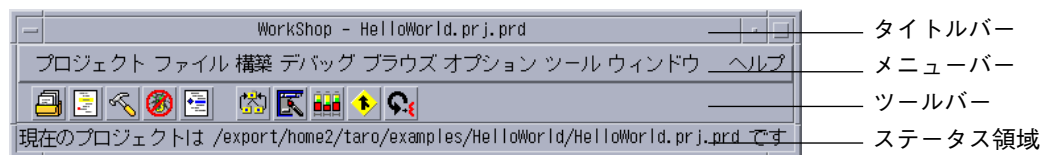


図 2-1 メインウィンドウ

メインウィンドウについての詳細は、オンラインヘルプを参照してください (オンラインヘルプを開くには、Sun WorkShop のメインウィンドウから「ヘルプ」▶「内容」を選択します)。Sun WorkShop のウィンドウで使用する色やフォントなどのリソースの設定については、付録 A を参照してください。

テキストエディタの選択とテキストエディタのオプション

Sun WorkShop 6 update 2 には、次の統合エディタが用意されています。

- NEdit (Sun WorkShop のデフォルトのエディタ)
- XEmacs
- GNU Emacs
- Vim (グラフィカルユーザーインターフェースオプション付き)
- Vi

デフォルトのエディタを変更したり、テキストエディタのオプションを設定したりするには、Sun WorkShop のメインウィンドウから「オプション」▶「エディタ用オプション」を選択して、「テキストエディタのオプション」ダイアログボックスを開きます。このダイアログボックスに表示されるオプションは、「使用するエディタ」プルダウンメニューで選択されているエディタによって異なります。「テキストエディタのオプション」ダイアログボックスから別のエディタを選択しない限り、現在選択しているエディタが Sun WorkShop のデフォルトのエディタになります。

各エディタのオプションについては、次を参照してください。

- エディタのメニューバーにある「ヘルプ」メニューからアクセスできるオンラインマニュアル
- オンラインヘルプ

Sun WorkShop の統合テキストエディタで使用する色やフォントなどのリソースを設定できます。CDE (共通デスクトップ環境) および CDE 以外の環境における Sun WorkShop のデフォルトのリソース設定の変更方法については、付録 A を参照してください。

注 - 日本語環境 (ja ロケール) で選択可能なエディタは、Vi、Mule (コマンド名 mule)、XEmacs (コマンド名 xemacs-mule) となります。

起動オプションとプロジェクトオプションの設定

Sun WorkShop 統合プログラミング環境のメインウィンドウの「オプション」メニューには、起動オプションとプロジェクトオプションがあります。

起動オプション

Sun WorkShop 統合プログラミング環境の起動時における次のような処理のデフォルトを設定できます。

- 前回の Sun WorkShop セッションのウィンドウのサイズと位置の復元
- 起動画面の表示
- 「Sun WorkShop へようこそ」ダイアログボックスの表示

これらの起動時オプションの設定を変更するには、メインウィンドウのメニューバーから「オプション」▶「起動オプション」を選択して、「起動オプション」ダイアログボックスを開きます。起動オプションについては、オンラインヘルプを参照してください。

プロジェクトオプション

Sun WorkShop 統合プログラミング環境のプロジェクトに対して次のような処理のデフォルトを設定できます。

- Sun WorkShop が起動されたとき、前回開かれていたプロジェクトを開き、そのプロジェクトに含まれていた項目を含むメニューピックリストを表示する。
- Sun WorkShop が終了されるか、別のプロジェクトが開かれたとき、プロジェクトの変更を自動的に保存するか、または保存するか破棄するかを問い合わせる。
- Sun WorkShop が終了されるか、別のプロジェクトが開かれたとき、自動的にすべてのメニューのメニューピックリスト項目を保存する。
- メニューピックリスト項目の最大数を 20 に設定する。
- ツールのデフォルトのディレクトリとして、自身が起動されたディレクトリを使用する。

これらのプロジェクトオプションの設定を変更するには、メインウィンドウのメニューバーから「オプション」▶「プロジェクトオプション」を選択して、「プロジェクトオプション」ダイアログボックスを開きます。プロジェクトオプションについては、オンラインヘルプを参照してください。

Sun WorkShop ツールの使用方法

メインウィンドウから、アプリケーションを作成、開発、デバッグ、チューニングするためのツールにアクセスできます。以降の章では、Sun WorkShop のツールと Sun WorkShop ツールを使用して、次の作業を行う方法について説明します。

- 開発プロジェクトの構築
- プログラムのデバッグ
- コードのブラウズ
- プログラムのパフォーマンス解析
- ソースファイルのマージ

第3章

プログラムの構築

Sun WorkShop プロジェクトでは、構築をカスタマイズしたり、メイクファイルを記述しないでプログラムを構築したり、自分で作成したメイクファイルを使用してプログラムを構築したりできます。また、「構築」ウィンドウと Sun WorkShop エディタを使用し、プロジェクトなしでアプリケーションを構築したり、一度に1つまたは複数の構築ジョブを実行したり、構築エラーを修正したりすることもできます。

具体的な手順とその他の情報については、オンラインヘルプを参照してください (オンラインヘルプには、Sun WorkShop のどのウィンドウからでも「ヘルプ」メニューまたは「ヘルプ」ボタンでアクセスできます)。

ターゲットの操作

Sun WorkShop における構築に関するターゲットは、次の2種類があります。

- Sun WorkShop ターゲット
- ユーザーメイクファイルターゲット

Sun WorkShop ターゲット

Sun WorkShop ターゲットは、構築ディレクトリ、構築コマンド、メイクファイル、構築ターゲットで構成されるオブジェクトです。

- **構築ディレクトリ**: 構築プロセスが起動されるディレクトリで、メイクファイルのデフォルトのディレクトリでもあります。
- **構築コマンド**: make ユーティリティを起動するコマンド。make ユーティリティはメイクファイルを読み取り、構築ターゲットを構築します。

- **メイクファイル**：構築ターゲットとファイル間の依存関係に基づいて構築ターゲットの更新方法を示す項目からなるファイルです。1つの依存関係が1つの構築ターゲットになるため、依存関係自体が複数の依存関係を持つことがあります。構築ターゲットを再構築するかどうかを決定するときに、`make` がトレースするツリー構造は、こうしたターゲットとファイル間の依存関係と下位依存関係によって形成されます。
- **構築 (make) のターゲット**：`make` が特定のメイクファイルに含まれる指示 (規則) からその構築方法に関する情報を得るオブジェクト。たとえば、`make` のターゲットは `all` になることもあれば、`clean` になることもあります。通常、メイクファイルは、最も一般的なターゲットがデフォルトターゲット (ターゲットを指定しなかったときに構築されるターゲット) になるように設計されています。

Sun WorkShop ターゲットは、構築されると「構築」メニューのピックアップリストと「構築」▶「ターゲットを編集」コマンドの Sun WorkShop ターゲットのピックアップリストに追加されます。構築を開始すると、Sun WorkShop ターゲットのリストの先頭にあるターゲットが構築されます。

プロジェクトには、複数のターゲットを含むことができます。実行可能ファイル、静的ライブラリ / アーカイブ、共有ライブラリ、Fortran アプリケーションの場合は、実行可能ファイル / ライブラリが1つのターゲットで、特殊な Clean ターゲットがもう1つのターゲットということになります (「構築」メニューのピックアップリストにある)。Clean ターゲットは、プロジェクトで作成されたファイル (たとえば .o ファイル)、ソースブラウザ用データベース、C++ テンプレートデータベース、実行可能ファイルそのもの、およびその他の構築関係のファイルのすべてを削除します。

複雑なプロジェクトの場合は、「構築」メニューのピックアップリストに複数のオブジェクトが含まれることがあります。たとえば、プロジェクトで、5つのライブラリと1つの実行可能ファイルを作成し、まとめてリンクできます。この結果、それぞれのライブラリおよび実行可能ファイルは1つの Sun WorkShop ターゲットになり、「構築」メニューのピックアップリストから選択することで個別に構築できます。

ユーザーメイクファイルターゲット

ユーザーメイクファイルターゲットは、`make` が特定のメイクファイルに含まれる指示 (規則) からその構築方法に関する情報を得るオブジェクトです。通常、メイクファイルは、最も一般的なターゲットがデフォルトターゲット (ターゲットを指定しなかったときに構築されるターゲット) になるように設計されています。

メイクファイルには、構築ターゲットとファイル間の依存関係に基づいて構築ターゲットの更新方法を表す項目が含まれます。1つの依存関係が1つの構築ターゲットであるため、依存関係自体が複数の依存関係を持つことがあります。構築ターゲットを再構築するかどうかを決定するときに、make がトレースするツリー構造は、こうしたターゲットとファイル間の依存関係と下位依存関係によって形成されます。

ユーザーメイクファイルプロジェクトの場合、「構築」メニューのピクリストに含まれるターゲットは、メイクファイルか構築するメイクファイルターゲットになります。

「構築」ウィンドウ

「構築」ウィンドウには、プログラムのコンパイルに関する情報が表示されます。「構築」ウィンドウは、Sun WorkShop メインウィンドウから「構築」▶「構築ウィンドウを表示」を選択して開きます。図 3-1 に、「構築」ウィンドウを示します。

「構築」ウィンドウから以下の操作を行うことができます。

- 構築の開始
- 構築の中止
- 構築パラメータの編集
- 構築出力の別ファイルへの保存
- 構築エラーの表示

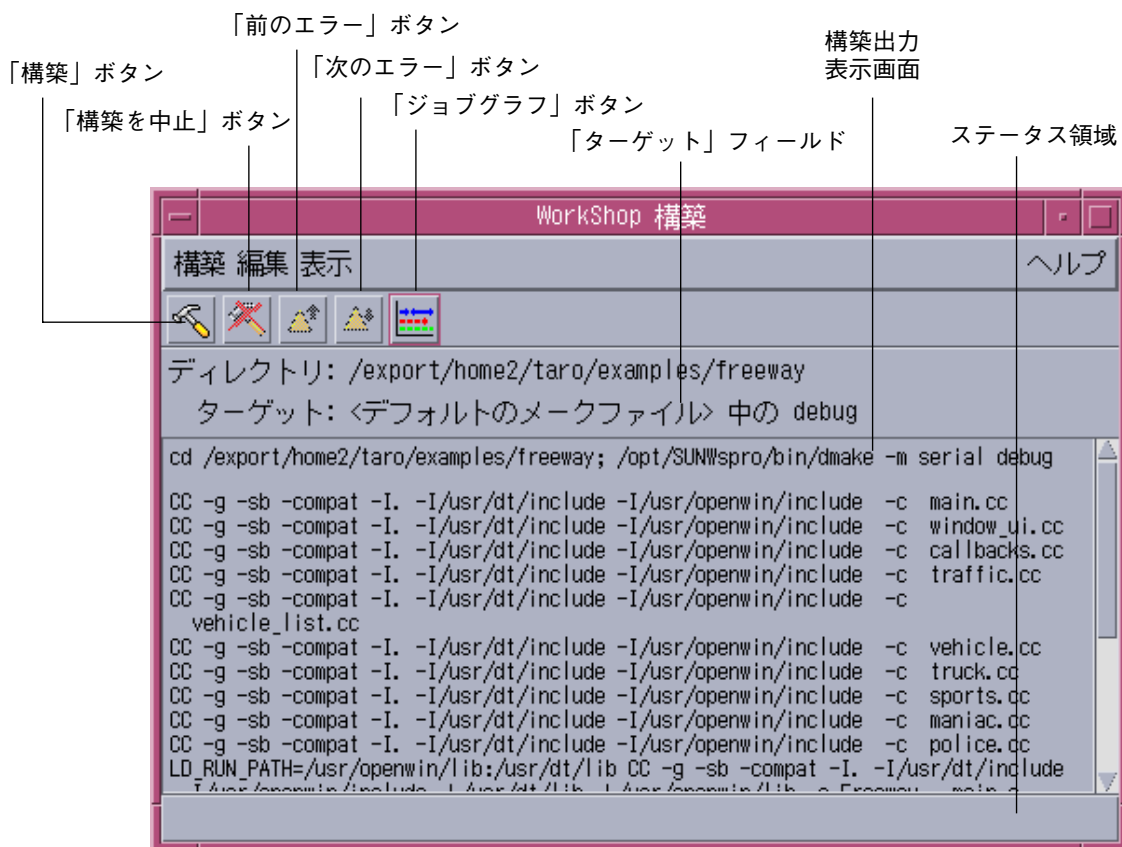


図 3-1 構築ウィンドウ

プログラムの構築

プロジェクトは、全体をまとめて構築することも、プロジェクトターゲットを1つだけ構築することもできます。プロジェクトターゲットを構築するように指示すると、Sun WorkShop 統合プログラミング環境は次のことを行ないます。

1. 「新規プロジェクト作成」ウィザードまたは「現在のプロジェクトを編集」ウィンドウで指定されたプロジェクト情報に基づいてメイクファイルを作成します。
2. make ユーティリティを起動します。

3. 「構築」ウィンドウを開いて、構築結果を表示します。

詳細は、オンラインヘルプを参照してください。

「ターゲットの新規定義」や「ターゲットの編集」ダイアログボックスで、構築パラメータを指定することもできます。新しい Sun WorkShop ターゲットを定義する場合は「ターゲットの新規定義」ダイアログボックス、既存の Sun WorkShop ターゲットを変更する場合は、「ターゲットの編集」ダイアログボックスをそれぞれ使用します(実際には、「ターゲットの新規定義」と「ターゲットの編集」ダイアログボックスは同じものです)。図 3-2 は、「ターゲットの新規定義」ダイアログボックスです。

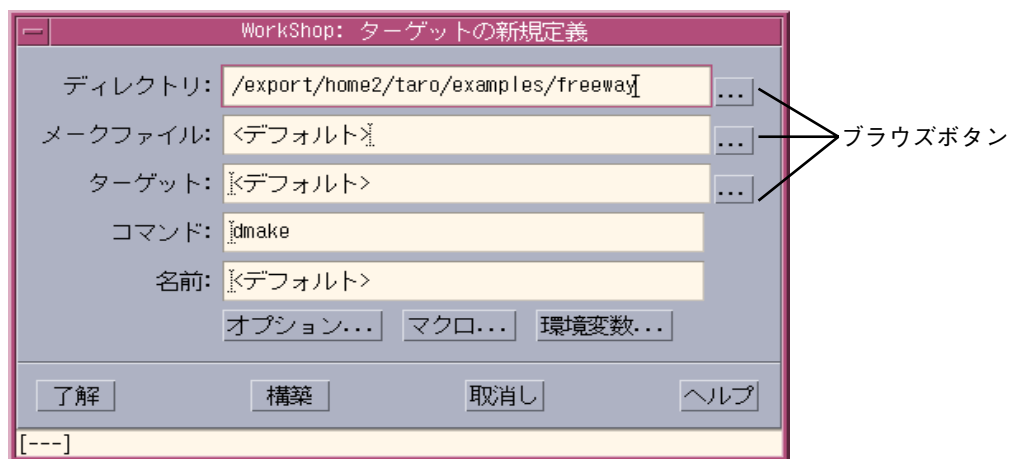


図 3-2 「ターゲットの新規定義」ダイアログボックス

デフォルト値を使用した構築

Sun WorkShop 統合プログラミング環境には、デフォルトの構築ターゲットとデフォルトの make コマンド (dmake) があります。このため、構築コマンドや構築ターゲットを指定しなくても構築を開始できます。ただし、ユーザーメークファイルプロジェクトを構築する場合や、プロジェクトが読み込まれていない場合は、メークファイルを指定する必要があります (Sun WorkShop 統合プログラミング環境は、makefile または Makefile という名前のファイルを検索して、make がどちらのファイルを使用するか決定できるようにします)。

このプロジェクト機能によって、「新規プロジェクト作成」ウィザードや「現在のプロジェクトを編集」ウィンドウからメークファイルを作成するように指示できます。

詳細は、オンラインヘルプを参照してください。

独自の構築値の指定

一意の名前を持つメイクファイル、独自の構築ターゲット、独自の構築コマンドがある場合は、「ターゲットの新規定義」ダイアログボックスや「ターゲットの編集」ダイアログボックスでその値を指定できます (この指定ができるのは、ユーザーメイクファイルプロジェクトに対してか、プロジェクトが読み込まれていない場合です)。たとえば、独自の構築コマンドを指定することによって、make の出力をフィルタにかけ、不要な警告を排除できます。make コマンドには、少なくとも構築ディレクトリを指定する必要があります。Sun WorkShop 統合プログラミング環境は指定された make コマンドを使用し、make の検索順序に従ってメイクファイルを検索します。詳細は、make(1S) のマニュアルページと、オンラインヘルプを参照してください。

構築オプションの指定

「構築オプション」ダイアログボックスで構築オプションを指定できます。「構築オプション」ダイアログボックスを開くには、「ターゲットの編集」ダイアログボックスから「オプション」をクリックします。使用できるオプションについての詳細は、「構築オプション」ダイアログボックスから「ヘルプ」をクリックしてください。必要なオプションの選択を終了したら、「構築オプション」ダイアログボックスで「了解」をクリックし、「ターゲットの編集」ダイアログボックスで「構築」をクリックします。

初めて分散メイクを行う場合は、dmake 構築サーバーとして参加するマシンを指定した .dmakerc 実行時構成ファイルを作成する必要があります。このファイルには、構築サーバーのグループ (リスト) と各構築サーバーに分散するジョブ数を含めます。dmake ユーティリティは、dmake ホストでこのファイルを検索することによって、ジョブを分散する場所の情報を取得します。一般に、.dmakerc 実行時構成ファイルは、作成した人のホームディレクトリに置きます。実行時構成ファイルを検索できなかった場合、dmake はローカルホストに 2 つのジョブを割り当てます。実行時構成ファイルの作成については、オンラインヘルプと dmake(1) のマニュアルページを参照してください。dmake ユーティリティについての詳細は、このマニュアルの付録 C と、dmake(1) のマニュアルページを参照してください。

マシンを構築サーバーとして設定するには、そのマシンのファイルシステムに /etc/opt/SPROdmake/dmake.conf という構成ファイルを作成する必要があります。このファイルが存在しない場合、dmake はそのマシンにジョブを割り当てるのを拒否します。dmake.conf ファイルには、そのマシン上で同時に実行可能なジョブ (全ユーザーからのジョブ) の最大数を指定します。詳細は、オンラインヘルプ、このマニュアルの付録 C、dmake(1) のマニュアルページを参照してください。

メイクファイルマクロの使用方法

「マクロ作成」ダイアログボックスにメイクファイルマクロを指定できます (このダイアログボックスを開くには、「ターゲットの編集」または「ターゲットの新規定義」ダイアログボックスから「マクロ」をクリックします)。メイクファイルマクロを使用することによって、記述ファイル内で使用されているファイルやコマンドオプションを参照できます。「マクロ作成」ダイアログボックスでは、Sun WorkShop ターゲットの「持続的構築マクロ」リストに対してメイクファイルマクロを追加、削除したり、マクロに対する値を変更したりできます。また、メイクファイルに現在定義されているマクロをリストに追加したり、その値を無効にすることもできます。詳細は、「マクロ作成」ダイアログボックスにある「ヘルプ」をクリックしてください。また、マクロの定義については、付録 B を参照してください。

環境変数の使用

「環境変数」ダイアログボックスで、構築に使用する環境変数を指定できます (このダイアログボックスを開くには、「ターゲットの編集」または「ターゲットの新規定義」ダイアログボックスから「環境変数」をクリックします)。「環境変数」ダイアログボックスでは、Sun WorkShop ターゲットの「持続的環境変数」リストに対して環境変数を追加、削除したり、環境変数に対する値を変更したりできます。構築を開始すると、環境変数を設定するための `setenv` コマンドが構築コマンドの前に付加されます。詳細については、「環境変数」ダイアログボックスにある「ヘルプ」をクリックしてください。

構築エラーの表示

構築で問題が発生すると、「構築」ウィンドウの構築出力表示区画に構築エラーメッセージが表示され、ソースファイル内のエラー発生場所へのハイパーテキストリンク (下線付きで強調表示されます) が作成されます (図 3-3 を参照)。エラーメッセージには、必ず、そのエラーを含むファイルの名前とエラーが発生した行番号、エラー内容が示されます。「構築」ウィンドウ内の下線の付いた部分をクリックすると、テキストエディタが起動して、そのエラーを含むソースファイルが表示されます。詳細は、オンラインヘルプを参照してください。

注 - 出力をハイパーテキストリンクに変換して生成できるのは、サンのコンパイラだけです。サンのコンパイラを呼び出さない構築コマンドを使用した場合、「構築」ウィンドウに表示された構築エラーからソースファイルへのリンクは作成されません。

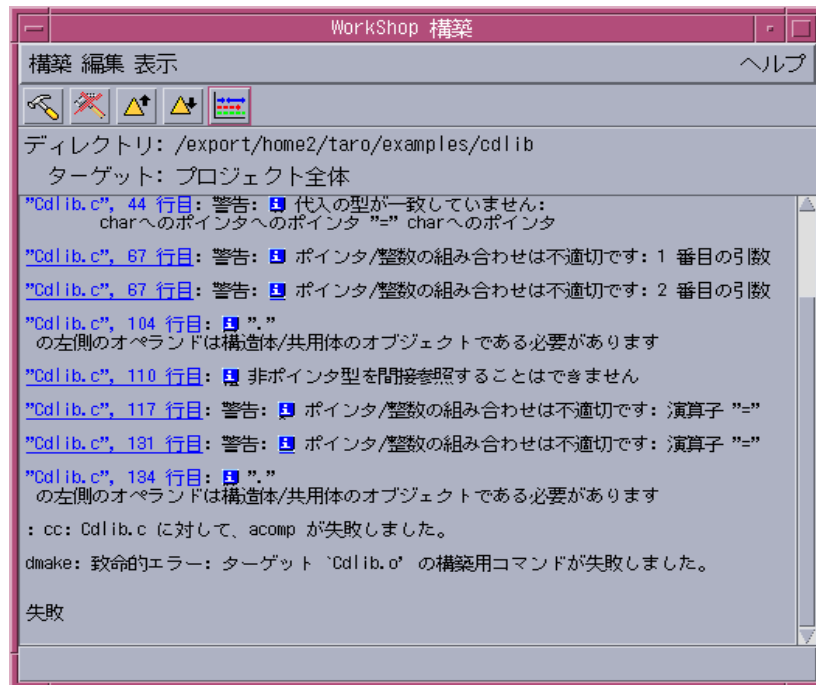


図 3-3 「構築」ウィンドウの構築出力表示区画に表示された構築エラーメッセージ

Fortran や C、C++ コンパイラによって出力されたエラーメッセージの構築エラーメッセージ部分には、情報アイコン ([i]) が表示されます。このアイコンをクリックすると、そのエラーに関する情報を示すポップアップウィンドウが表示されます (図 3-4 を参照)。

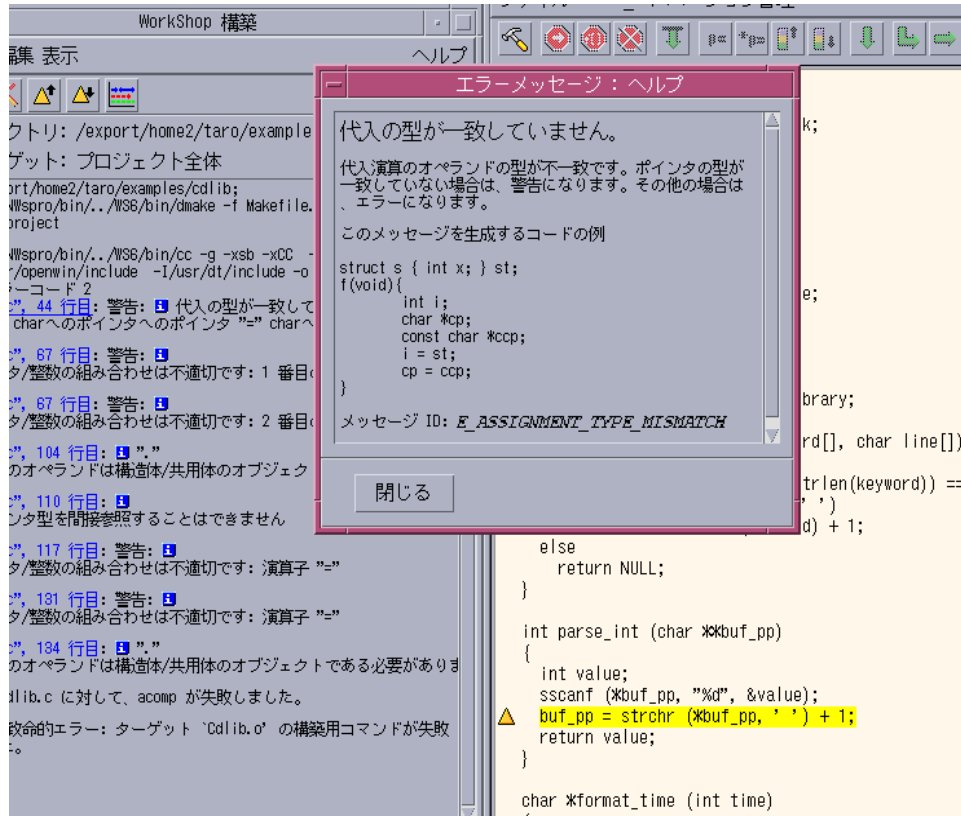


図 3-4 構築エラーと、エラーに関する情報を示すダイアログボックス

構築の終了

現在の構築プロセスを強制終了して、すべての構築ウィンドウを閉じるには、「構築」ウィンドウから「構築」▶「構築終了」を選択します。

現在の構築処理を強制終了しないで構築ウィンドウを閉じるには、「構築」▶「閉じる」を選択します。

第4章

プログラムのデバッグ

Sun WorkShop 統合プログラミング環境には、プログラムの実行を制御し、停止したプログラムの状態を検査するデバッグウィンドウが用意されています。プログラムを動的に実行して完全に制御し、パフォーマンスデータを収集できます。

「デバッグ」ウィンドウ (「デバッグ」▶「デバッグウィンドウの表示」を選択して開きます) と、その下位のデバッグウィンドウから次のことを行えます。

- デバッグセッションの開始 (一度に複数セッションの開始が可能)
- プログラムを実行停止位置の指定
- プログラムの実行制御
- ブレークポイントの利用
- プロセスへの接続
- コードのトレース
- 式と変数の評価
- 呼び出しスタックの利用
- プログラムの修正
- マルチスレッドプログラムのデバッグ
- パフォーマンスデータの収集
- 実行時検査機能の利用
- 配列のグラフ表示
- デバッグ環境の設定
- カスタムボタンの作成

また、Sun WorkShop には、コードのデバッグに役立つマシンレベルのコマンドやその他のコマンドも用意されています。「dbx コマンド」ウィンドウから標準の dbx コマンドを利用することもできます。

デバッグ方法とデバッグの概念、デバッグウィンドウ、dbx コマンドについては、オンラインヘルプを参照してください (オンラインヘルプには、Sun WorkShop のどのウィンドウからでも「ヘルプ」メニューまたは「ヘルプ」ボタンでアクセスできます)。

デバッグの準備

デバッグを行うには、ソースファイルをコンパイルしたときのデバッグ情報が必要になります。デバッグ情報を作成するには、次のいずれかを行います。

- メインウィンドウから「プロジェクト」▶「プロジェクトの編集」を選択して、「プロジェクトの設定の変更」をクリックします。必要な選択を終了したら、「了解」をクリックして、プロジェクトを構築します。詳しくは、「現在のプロジェクトを編集」ウィンドウから「ヘルプ」をクリックするか、オンラインヘルプを参照してください。
- -g または -g0 (0 は数字のゼロ) オプションを使用してコンパイルする。これらのオプションは、コンパイル中にデバッグ情報を生成するようコンパイラに指示します (メークファイル内にこれらのオプションを指定する方法については、付録 B を参照)。プログラムのデバッグの準備についての詳細は、『dbx コマンドによるデバッグ』を参照してください。

デバッグの開始

プログラムのデバッグを開始するには、次の手順に従ってください。

1. デバッグの状態を選択します。
 - 「デバッグ」▶「高速実行モード」: プログラムを普通に実行しますが、デバuggはプログラムが異常終了した場合、プログラムを救出できるようにバックグラウンドで動作します。すでにプログラムのデバッグ作業が終了していて、シンボル情報の読み込みを行う必要がなく、プログラムに対して修正した内容が正しく動作するかをテストするだけのテストをする場合には、このモードを使用してください。

- 「デバッグ」 ▶ 「デバッグモード」 (またはメインウィンドウのツールバーから「デバッグ」 ボタンをクリック): デバッグサービスの全機能を使用してプログラムをデバッグする場合は、このモードを選択します。

2. デバッグするプログラムを選択します。

現在のプログラムをデバッグするには、メインウィンドウのツールバーから「デバッグ」 ボタンをクリックします。別のプログラムをデバッグするには、次のいずれかの操作を行います。

- 以前に Sun WorkShop 統合プログラミング環境で実行またはデバッグしたプログラムをデバッグする場合

「デバッグ」 メニューのピックリストからプログラムを選択します。

- ピックリストにないプログラムをデバッグする場合

「デバッグ」 ▶ 「新規プログラム」 を選択して新しいプログラムを読み込みます。

- 実行中の別のプロセスを接続する場合

「デバッグ」 ▶ 「プロセスを接続」 を選択します。

- プログラム実行の失敗が原因で生成されたコアダンプファイルをデバッグする場合

「デバッグ」 ▶ 「コアファイルの読み込み」 を選択します。

プログラムが読み込まれると、「デバッグ」 ウィンドウ (図 4-1 を参照) とテキストエディタのウィンドウが開きます。テキストエディタのウィンドウにプログラムのソースコードを表示して、編集や基本的なデバッグ操作を行うことができます。エディタウィンドウのツールバーには、よく使用するデバッグ機能、特にソースのコンポーネントを引数として使用する機能のボタンと、Sun WorkShop の他のツールのボタンがあります。

デフォルトのテキストエディタの変更と、テキストエディタのオプションの設定については、11 ページの「テキストエディタの選択とテキストエディタのオプション」を参照してください。

3. 「デバッグ」 ウィンドウから次のいずれかの操作を行い、プログラムを実行します。

- F6 キーを押す。
- 「デバッグ」 ウィンドウのツールバーから「開始」 をクリックする。
- 「デバッグ」 ウィンドウから「実行」 ▶ 「開始」 を選択する。
- 「デバッグ」 ウィンドウのツールバーから「継続」 をクリックする。

- 「デバッグ」ウィンドウから「実行」▶「継続」を選択する。

「デバッグ」ウィンドウについては、オンラインヘルプを参照してください。

デバッグセッション中に、プログラム引数や実行ディレクトリ、環境変数などの実行時パラメータの設定を変更できます。詳細は、オンラインヘルプを参照してください。

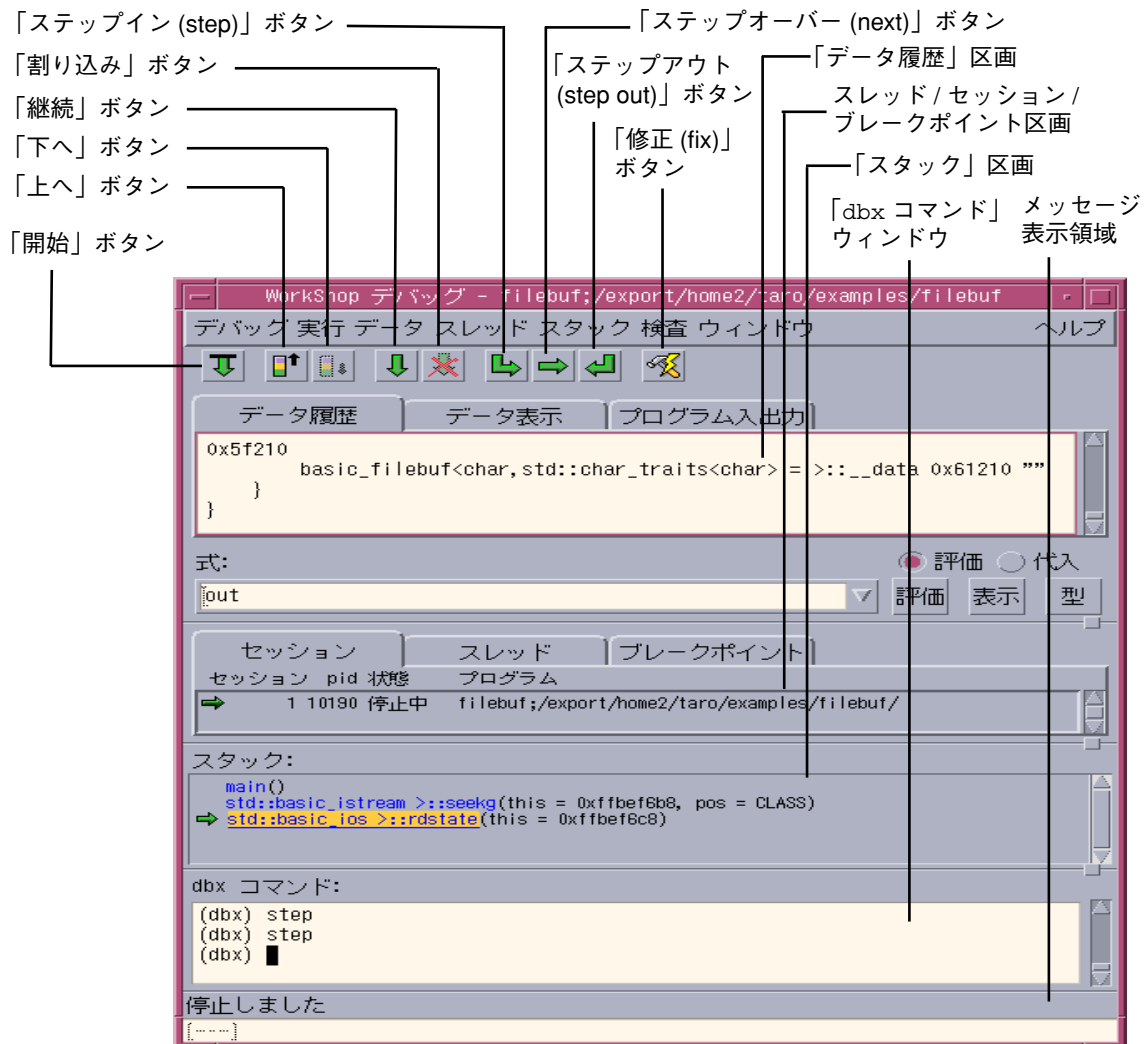


図 4-1 Sun Workshop の「デバッグ」ウィンドウ

デバッグウィンドウのカスタマイズ

次のオプションのデフォルト値を変更することによって、デバッグウィンドウをカスタマイズできます。

- デバッグの出力
- デバッグの動作
- ウィンドウのレイアウト
- ウィンドウの動作
- 「データ表示」ウィンドウ
- 言語とスコープ
- 実行時検査
- データグラフ
- デバッグのパフォーマンス
- フォークとスレッド
- コマンド行のみ
- 上級オプション

デバッグオプションを表示するには、「デバッグ」ウィンドウから「デバッグ」▶「デバッグオプション」を選択します。「デバッグオプション」ダイアログボックスでは、デバッグウィンドウのデフォルト値を変更することができます。dbxenv コマンドを使用して、dbx の環境変数を設定することによって、デフォルト値の多くを変更することもできます (詳細はオンラインヘルプを参照)。

デバッグオプションの設定についての詳細は、次を参照してください。

- オンラインヘルプ
- 『dbx コマンドによるデバッグ』

コードのステップ実行

コードを 1 行ずつ移動して、ステップごとに表示しながら実行できます。ステップ実行すると、プログラム内の現在の位置が緑色で強調表示されます。これをプログラムカウンタ (PC) と呼びます。PC は、各ステップごとに次に実行されるソース行に移動します。

ステップ実行には、次の 3 つの方法があります。

ステップイン (step)	ソース行を 1 行先に進めます。ソース行が関数呼び出しの場合、デバッグサービスは関数の最初の文の前で停止します。
ステップオーバー (next)	ソース行を 1 行先に進めます。ソース行が関数呼び出しの場合、デバッグサービスは関数の命令についてはステップ実行しないで、関数全体を実行します。
ステップアウト (step out)	現在の関数の実行を終了し、その関数呼び出しの直後のソース行で停止します。PC が呼び出しと同じソース行で停止した場合は、その呼び出しに関する機械命令がいくつか残っています。もう一度ステップ実行を行うと、呼び出しが完了し、次のソース行に進みます。

コードのステップ実行についての詳細は、オンラインヘルプを参照してください。

ブレークポイントの設定

ブレークポイントを設定することによって、デバッグウィンドウでの実行を停止できます。単純なブレークポイント設定では、実行を停止する特定のコード行位置や手続き、関数を指定できます。詳細なブレークポイント設定では、C++ クラス内や特定の条件、特殊なイベントでの停止を指定したり、データの変化を追跡するように指定したり、独自のブレークポイントを作成したりできます。

ブレークポイントの設定と解除は、エディタのウィンドウまたは「ブレークポイント」ウィンドウで行うことができます。エディタのウィンドウでは、コード行位置や関数に対するブレークポイントを設定、解除できます。「ブレークポイント」ウィンドウでは、シグナルが発生したときなどの、より複雑なブレークポイントを設定できます（「デバッグ」ウィンドウの「ブレークポイント」タブには、すでに設定されているブレークポイントが表示されます）。

ブレークポイントの設定と使用方法については、オンラインヘルプを参照してください。

値とデータの検査

評価とは、式の値に対して一度だけ行われる無作為抽出検査です。エディタウィンドウまたは「デバッグ」ウィンドウから、いつでも式を評価できます。「デバッグ」ウィンドウの「データ表示」タブまたは独立した別の「データ表示」ウィンドウを使用して、プログラム停止時の値の変化を追跡できます。

評価結果は、「デバッグ」ウィンドウの「データ履歴」タブに一覧表示されます。点線は、前回の評価から評価コンテキストに変化があったことを示します。「データ履歴」タブの履歴リストは、以前に評価された式のリストです。「データ履歴」タブの履歴リストは、「データ」▶「履歴消去」を選択することによっていつでも消去できます。

エディタのウィンドウから式を評価するには、次のいずれかを行います。

- 式のパルーン評価機能を利用する。エディタの現在のマウスカーソル位置にある式の現在値がただちに表示されます。詳細は、オンラインヘルプを参照してください。
- ソース表示から評価する変数または式を選択して、次のいずれかを行う。
 - 「評価」をクリックするか、「WorkShop」▶「評価」▶「選択」を選択する (選択した式の値を求める場合)。
 - 「*を評価」をクリックするか、「WorkShop」▶「評価」▶「ポインタとして」を選択する (ポインタ型の式のポイント先を評価する場合)。

値は「データ履歴」タブに表示されるか、エディタのウィンドウ最下部のメッセージ表示領域に表示されます (評価結果が短い場合)。評価コンテキストに変化があると、必ず、「データ履歴」タブのリストに区切り線が挿入されます。詳細は、オンラインヘルプを参照してください。

データ値の監視

デフォルトでは、「データ表示」は「デバッグ」ウィンドウのタブとして表示されますが、独立したウィンドウとして表示されるように設定することもできます。詳細は、オンラインヘルプを参照してください。

「デバッグ」ウィンドウの「データ表示」タブから、プログラム実行中の式の値の変化を観察できます。選択した一群の式は、ブレークポイント位置やステップ実行位置、プログラムの実行が中断されたときなどの、プログラムの実行停止のたびに、自動的に評価されます。式の値が変化すると、その値が太字で強調表示されます。

「データ表示」には、実行を中止するたびに値の変化が表示されます。プログラムの実行中に値の変化を観察する必要がある場合は、「アクセス時」ブレークポイント機能を使用してください(詳細は、オンラインヘルプを参照)。このブレークポイント機能を使用して、特定のメモリー位置に対する読み取りまたは書き込みが行われるたびにプログラムの実行を中止するように指示できます。

「データ表示」タブまたはウィンドウから、ポップアップウィンドウを開いて、式に関する追加情報を表示して管理できます。

詳細は、オンラインヘルプを参照してください。

パフォーマンスデータの収集

デバッグウィンドウでのプログラムの実行中に、標本コレクタを使用して、パフォーマンスデータを収集し、パフォーマンスアナライザで利用できる実験ファイルに出力できます。標本コレクタは、クロックに基づくプロファイルデータやスレッド同期の待ち状態トレースデータ、ハードウェアカウンタのオーバーフローに関するプロファイルデータ、アドレス空間データを収集できます。コレクタは、ページフォルトや入出力データ、コンテキストスイッチ、ワークセットやページング統計情報などの、グローバル実行の統計情報を自動的に記録します。

パフォーマンスデータの収集については、次を参照してください。

- このマニュアルの第 6 章
- 『プログラムのパフォーマンス解析』
- オンラインヘルプ

実行時エラーの検出

実行時検査 (RTC) では、開発段階におけるアプリケーションの実行時エラーを自動検出できるほか、次のことが行えます。

- メモリーアクセスエラーの検出
- メモリーリークの検出
- メモリーの使用状況に関するデータの収集
- すべての言語への対処
- ライブラリなど、ソースがないコードの処理

実行時検査を行う場合は、プログラムを実行する前に検査の種類を選択しておく必要があります。その後、プログラムを実行すると、メモリーの使用状況に関するレポートが作成されます。

詳細は、オンラインヘルプを参照してください。

コードのトレース

トレース機能は、プログラムで発生するイベントに関する情報を収集して、収集したデータを「dbx コマンド」ウィンドウに表示します。このとき、プログラムの実行は停止しません。

トレースにフィルタを設定しないと、実行直前のソースコードの各行が表示されるため、単純なプログラム以外では大量の情報が出力されます。トレースにフィルタを設定して、トレースするプログラムのイベントを限定すると、特定の種類のデータを表示できます。たとえば、特定の関数呼び出しや特定の名前のメンバー関数、クラス内の関数、または関数からの終了をトレースできます。変数の値の変化をトレースすることもできます。

イベントは、デバッグ対象に設定されたプログラムのイベントです。代表的なイベントとしては、特定の変数の値の変化があります。ハンドラはデバッグイベントを管理します。「ブレイクポイント」ウィンドウのトレースリストは、イベントの一種であるトレースを管理するため、トレースハンドラと呼ばれます。

詳細は、オンラインヘルプを参照してください。

呼び出しスタックの検査

呼び出しスタックは、現在アクティブなすべてのルーチン、すなわち、呼び出されたが、まだそれぞれの呼び出し元には復帰していないすべてのルーチンを表します。スタック区画には、関数とその引数が、呼び出された順序で一覧表示されます。初期関数(C および C++ プログラムでは `main()`) はスタック区画の先頭、プログラムが停止した時点で実行されていた関数はスタック区画の最後に示されます。後者の関数を「停止位置関数」と呼びます。

停止位置関数のソースコードは、エディタのウィンドウに表示され、次に実行される行は緑色で強調表示されます。

呼び出しスタックでは、次のいずれかの操作を行うことができます。

- 「上へ」 ボタンをクリックするか「スタック」 ▶ 「上へ」 を選択して、スタック内を 1 レベル上に移動する。
- 「下へ」 ボタンをクリックするか「スタック」 ▶ 「下へ」 を選択して、スタック内を 1 レベル下に移動する。
- 戻りたいフレームの横にカーソルを置いて「スタック」 ▶ 「現フレームまでポップ」を選択し、複数のフレームを削除する。
- 「スタック」 ▶ 「ポップ」を選択して、停止した関数をスタックから削除する
- 戻りたいフレームの横にポインタを置き、「スタック」 ▶ 「現フレームまでポップ」を選択し、複数のフレームを削除する。

「ポップ」は、制限付きの元に戻す機能です。再び現在の関数の先頭から実行を開始するには、親スタックフレームまでポップし、その関数にステップインします。これで、関数の先頭に戻ります。

呼び出しスタックの使用方法については、オンラインヘルプを参照してください。

マルチスレッドプログラムのデバッグ

マルチスレッドプログラムであることが検出されると、「デバッグ」ウィンドウの「スレッド」タブが開きます。この状態から、「セッション」タブをクリックすることによって、セッションを表示できます。マルチスレッドプログラムの場合、「セッション」タブには、現在の選択されているプロセス内のスレッドに関する情報が一覧表示されます。現在のスレッドには、緑色の矢印が付きます。

詳細は、オンラインヘルプを参照してください。

複数プロセスの同時デバッグ

各プログラムをそれぞれ独立したデバッグセッションに接続し、一度に複数のプログラムをデバッグできます。この機能は、たとえば次のような場合に使用します。

- プロセスと、そのプロセスがフォークする子プロセスのデバッグ
- クライアントとサーバープログラム
- 関連する2つのプログラム

1つのプログラムをデバッグしているときに、別のプログラムをデバッグしようとするすると、プログラムのデバッグ中であることを示すメッセージが表示され、現在のセッションを終了または切り離して、新しいデバッグセッションを読み込むか、または現在のセッションを再利用するか、あるいは両方のセッションをデバッグするか選択するよう求められます。両方のプログラムを同時にデバッグする場合にだけ両方のセッションのデバッグを選択してください。

セッションの管理

「デバッグ」ウィンドウの「セッション」タブおよび「活動状態セッション」ダイアログボックスは、現在アクティブな、すべてのデバッグセッションの一覧です。「活動状態セッション」ダイアログボックスを開くには、「デバッグ」ウィンドウから「デバッグ」▶「セッション管理」を選択します。現在のプログラムには、矢印が付きます。

複数のセッションをデバッグするとリソースが消費され、システムの処理速度が低下することがあります。「デバッグ」ウィンドウには、現在アクティブなセッション数が表示されます。不要になったセッションは、「活動状態セッション」ダイアログボックスにある「切り離す」または「セッション終了」ボタンをクリックして、セッションを削除してください。

複数のセッションをデバッグすることはできますが、同時に複数のセッションのコンテキストを表示することはできません。別のセッションに切り替えると、「デバッグ」ウィンドウ、エディタウィンドウ、「dbx コマンド」ウィンドウなどはすべて、切り替え先のセッションのコンテキストを反映した内容になります。

2つのプログラムを横に並べて表示する(それぞれのプログラムにエディタのウィンドウと「デバッグ」ウィンドウを表示する)には、2つの WorkShop アプリケーションを実行する必要があります。2つの WorkShop アプリケーションが同じエディタを使用しないようにするには、両方の WorkShop アプリケーションを次のコマンドを使用して起動します。

```
% workshop -s editsessionname
```

たとえば、最初の Sun WorkShop アプリケーションは `workshop -s 1`、2つ目の Sun WorkShop アプリケーションは `workshop -s 2` のようにします。詳細は、`workshop(1)` のマニュアルページを参照してください。

セッションの管理については、オンラインヘルプを参照してください。

子プロセスのデバッグ

プロセスが子プロセスをフォークした場合は、親または子プロセスのみ、またはその両方をデバッグするかを選択できます。通常、フォークされたプロセスからはすべてのブレークポイントが削除されますが、この設定を無効にすることもできます。

詳細は、オンラインヘルプを参照してください。

デバッグの終了

現在 Sun WorkShop の制御下にあるプロセスを終了しないで「デバッグ」ウィンドウを閉じるには、「デバッグ」▶「閉じる」を選択します。この場合、Sun WorkShop の制御下にあるプロセスは、メモリーと CPU 時間を消費しながら、動作を継続します。Sun WorkShop は、それらプロセスに関するデータの記録を継続します。

Sun WorkShop の制御下にあるすべてのプロセスを終了して、すべてのデバッグウィンドウを閉じるには、「デバッグ」ウィンドウから「デバッグ」▶「デバッグを終了」を選択します。

第5章

ソースコードのブラウズ

Sun WorkShop 統合プログラミング環境では、C、C++、Fortran 77、Fortran 95 ソースコードをブラウズするためのモードとして、パターン検索とソースブラウズの2つのモードが用意されています。パターン検索モードでは、ソースコードから、コメント内のテキストなどの任意の文字列を検索できます。ソースブラウズモードでは、プロジェクトを作成または編集するときにソースブラウズ情報を作成するようにプロジェクトザードで指定することによって作成されたソースブラウズ用データベースを使用して、ソースコードから任意のプログラム定義シンボルに一致するものをすべて検索できます。問い合わせに一致するものは、その前後のソースコードとともに、「ブラウズ」ウィンドウの一致区画に表示されます。

プログラム内の関数やサブルーチンの関係をグラフ表示することもできます。ソースコードが C++ で書かれている場合は、プログラムに定義されているクラスをブラウズして、グラフ表示できます。

具体的な手順とブラウズについての詳細は、オンラインヘルプを参照してください(オンラインヘルプには、Sun WorkShop のどのウィンドウからでも「ヘルプ」メニューまたは「ヘルプ」ボタンでアクセスできます)。

パターン検索モード

パターン検索機能は、現在のディレクトリまたは `sb_init` ファイルに指定されているインポートディレクトリから、コメント内のテキストなどの任意の文字列を検索します(`sb_init` ファイルと複数ディレクトリの検索については、オンラインヘルプを参照)。

パターン検索機能は、次のような場合に使用します。

- 単純に文字列を素早く検索する場合。
- 検索するディレクトリ内にソースブラウザ用データベースがある場合。
- 関数の呼び出し関係やクラスの階層構造をグラフ表示しない場合。
- クラスのデータやメンバー関数を調べない場合。

パターン検索機能は、grep 構文を利用し、「ブラウザ」ウィンドウの「パターン」フィールドに入力された文字列に一致するものを探します (図 5-1 を参照)。詳細は、オンラインヘルプを参照してください。

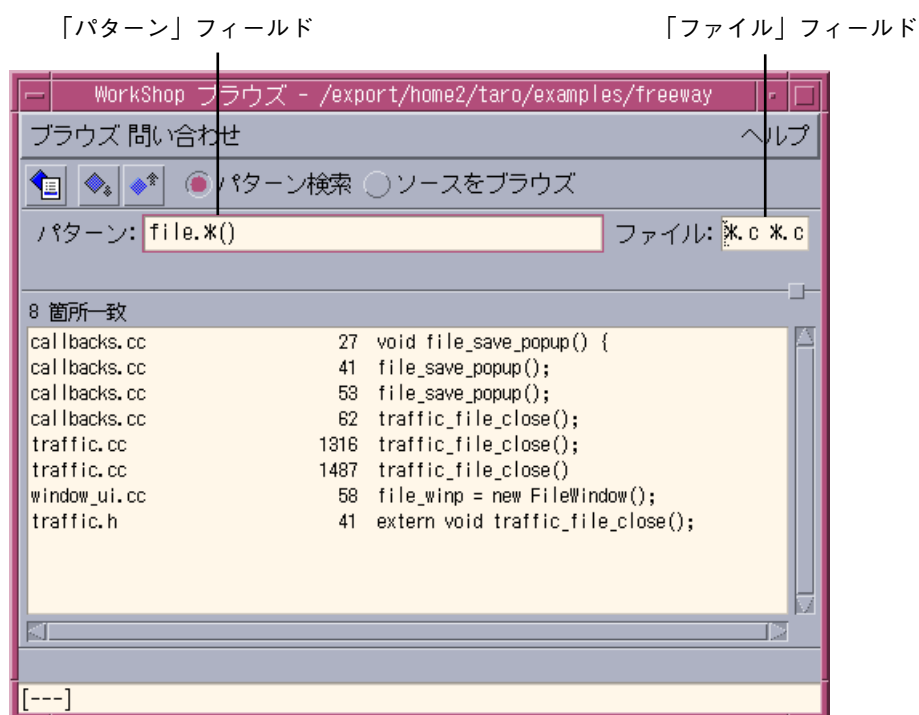


図 5-1 パターン検索モードでの「ブラウザ」ウィンドウ

パターン検索で使用できる特殊文字

「パターン」フィールドでは、ソースコードに入力されているとおりに文字列を入力することも、特殊文字(ワイルドカード)文字を使用してパターンを指定することもできます。表 5-1 に、「ブラウザ」ウィンドウの「パターン」フィールドで使用できる特殊文字をまとめます。

表 5-1 パターン検索で使用できる特殊文字

文字	意味	例
ピリオド(.)	任意の1文字	l.nes は lanes、lines のどちらにも一致
アスタリスク(*)	*の前に指定された文字を含む任意の文字列(パターンの最初の文字として使用された場合を除く)	file*() は file を含み、任意の文字と () を伴う文字列に一致 (traffic_file_close() など)。*file は、file で始まる文字列にのみ一致
サーカムフレックス (^)	行頭で一致	^tr* は、traffic、truck などの tr で始まる行に一致
ドル記号(\$)	行末で一致	lanes\$ は、lanes で終わる行に一致
バックスラッシュと左小なり括弧(\<)	語頭で一致	\<get は get_foobar に一致し、widget には一致しない。
バックスラッシュと大なり括弧(\>)	語尾で一致	\<String\> は String *foo に一致し、XmStringCreate() には一致しない。

サーカムフレックスとドル記号で問い合わせ内容を囲むと、その問い合わせ内容に行全体が一致するものが検索されます。

詳細は、オンラインヘルプを参照してください。

複数ディレクトリの検索

パターン検索では、sb_init テキストファイルを使用し、複数のディレクトリに分散しているソースファイルに対して検索を行うことができます。具体的な手順については、オンラインヘルプを参照してください。

ソースブラウザモードの使用法

ソースブラウザモードでは、問い合わせが発行されると、ブラウザ中のソースファイルに関する情報が格納されているデータベースを使用して検索が行われます。ソースブラウザモードは、次のような場合に利用します。

- ソースブラウザ用データベースがある場合。
- 関数、クラス、構造体、共用体、レコード、あるいはそれらの使用方法や定義、または割り当て箇所などの言語要素を検索する場合。
- 関数の呼び出し関係やクラスの階層構造をグラフ表示する場合。
- クラスのデータ関数やメンバー関数を検査する場合。

図 5-2 は、ソースブラウザモードでの「ブラウザ」ウィンドウを示しています。

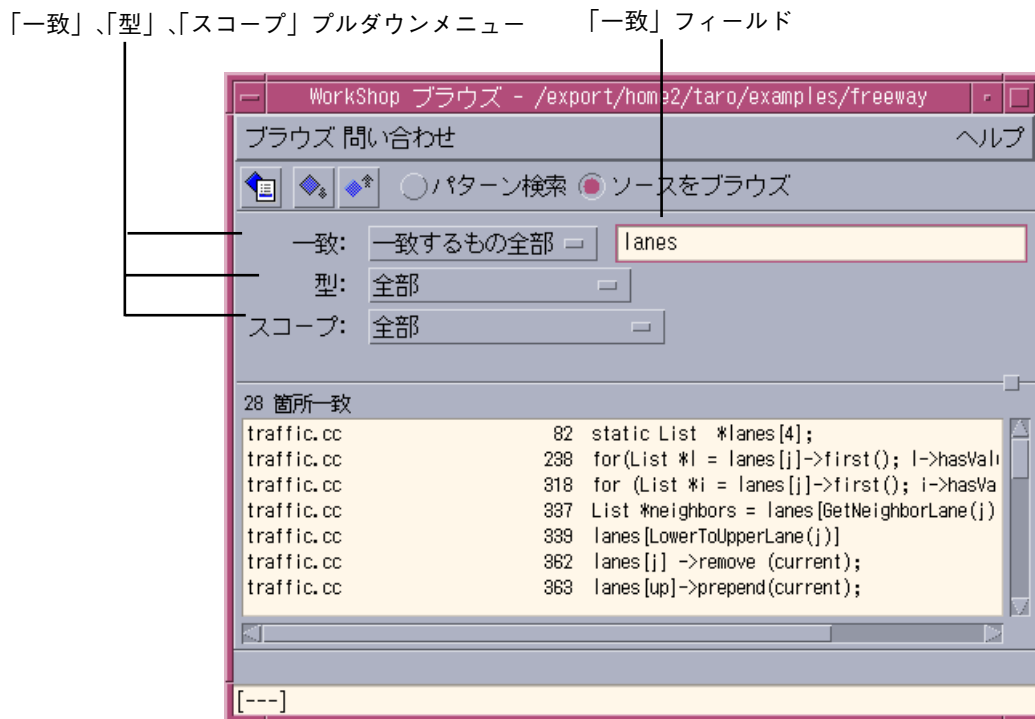


図 5-2 ソースブラウザモードのときの「ブラウザ」ウィンドウ

ソースブラウザ用データベース

Sun WorkShop 統合プログラミング環境は、プログラムの静的構造に関する情報が含まれているデータベースからブラウザ情報を取得します。ブラウザがどのように機能するかは、アクセスするデータベースによって異なります。Sun WorkShop には、次のデータベースオプションがあります。

- データベースなし：データベースがない場合は、ソースブラウザモードではなく、パターン検索モードを使用する必要があります。詳細は、オンラインヘルプを参照してください。
- コンパイラ生成ブラウザデータベース：このデータベースは、全ブラウザ機能を利用することを可能にします。ソースブラウザモードでは、このデータベースを検索して、問い合わせに応答します。

プロジェクトを作成または編集するときにプロジェクトウィザードからコードをコンパイルするときにデータベースを生成することを選択できます。あるいは、メタファイルに適切なソースブラウザオプションを付けてソースファイルを構築することによってデータベースを生成することもできます。具体的な手順については、オンラインヘルプを参照してください。

- タグ生成データベース：このデータベースは、コンパイルをしないでソースファイルをブラウザすることを可能にします。関数や大域変数に対する問い合わせを行なって、関数呼び出しの情報を表示できます (ただし、グラフ表示機能は利用できません)。タグ生成データベースでは、変数や型、関数の大域定義だけが検出され、関数呼び出しに関する情報が収集されます。C++ のメンバーに対する関数呼び出しは、そのメンバーが明示的に呼び出されている場合にのみ検出されます。具体的な手順については、オンラインヘルプを参照してください。

ソースブラウザで使用できる特殊文字

「一致」フィールドでは、ソースコードに入力されているとおりに名前や関数を入力することも、特殊文字 (ワイルドカード) 文字を使用して文字列を指定することもできます。

表 5-2 に、「ブラウザ」ウィンドウの「一致」フィールドで使用できる特殊文字をまとめます。

表 5-2 ソースブラウザで使用できる特殊文字

文字	意味	例
ピリオド (.)	任意の 1 文字	.ehicle は、vehicle、Vehicle のどちらにも一致。
アスタリスク (*)	* の前に指定された文字列を含む任意の文字数の文字列	vehi* は、vehicle_length() などの vehi で始まる任意の文字列に一致。 vehi* は、veh. に一致するが、vehicle_length() には一致しない。

詳細は、オンラインヘルプを参照してください。

複数ディレクトリのブラウザ

ユーザーメークファイルプロジェクトを除く全プロジェクトの場合：プロジェクトを作成または編集するとき、コンパイル中にソースブラウザ用の情報を作成するようにプロジェクトウィザードで指定すると、すべてのブラウザディレクトリがマージされます。このようにして、パターン検索またはソースブラウザモードで検索すると、プロジェクト内のすべてのソースディレクトリが自動的に検索されます。詳細は、オンラインヘルプを参照してください。

ユーザーメークファイルプロジェクトの場合：ソースファイルが複数のディレクトリに分散している場合は、通常は、それぞれのディレクトリでコンパイラを実行することになります。デフォルトでは、コンパイラはそれらのディレクトリごとにソースブラウザ用データベースを作成します。ソースブラウザは一度に 1 つのデータベースしかブラウザしないため、検索されるのは、現在のディレクトリに存在するアプリケーション部分だけです。このデフォルトの動作は、データベースをインポートすることによって変更できます。データベースをインポートする具体的な手順については、オンラインヘルプを参照してください。

ブラウザとグラフ表示の関係

図 5-3 は、「ブラウザ」ウィンドウと「コールグラフ」ウィンドウ、「クラスグラフ」ウィンドウ、「クラスブラウザ」ウィンドウの相互関係を示しています。

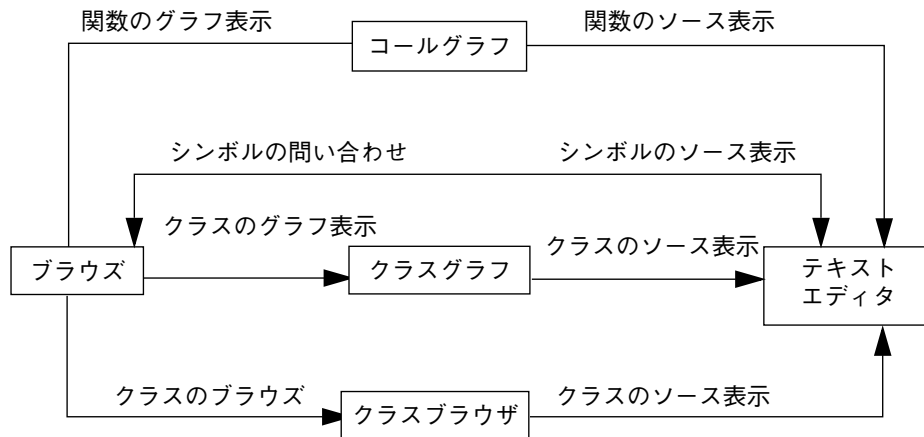


図 5-3 ブラウズ、クラスグラフ、クラスブラウザの相互関係

関数のグラフ表示

「コールグラフ」ウィンドウでは、ANSI C、C++、Fortran で書かれたプログラム内の関数の関係をグラフ表示できます。1 つ以上の関数を選択して、その呼び出し元と呼び出し先の関数を表示できます。「コールグラフ」ウィンドウは、関数とサブルーチンの呼び出し関係をグラフィカルに表示する手段です。具体的な使用手順とその他の情報については、オンラインヘルプを参照してください。

関数の関係を表示するには、ソースブラウザ用データベースが必要です (43 ページの「ソースブラウザ用データベース」を参照)。

注 – Sun WorkShop 統合プログラミング環境では、仮想関数の関係もグラフ表示できますが、実際に呼び出される関数を特定することはできません。たとえば、main が b::d() という仮想関数を呼び出し、この仮想関数が実際には b1::d() または b2::d() のいずれも呼び出せる場合、Sun WorkShop 統合プログラミング環境が呼び出される関数を特定することはできません。グラフのウィンドウには、main が b::d() を呼び出す関係は示されますが、main と b1::d() や main と b2::d() の関係は示されません。

「コールグラフ」ウィンドウ内のノードの背景、コールグラフ区画の背景、ノードのボックスの枠、ノードのボックス内のテキスト、ノード間の矢印に使用する色を変更するには、WORKSHOP リソースファイルを編集します (69 ページの「「コールグラフ」ウィンドウと「クラスグラフ」ウィンドウの色」を参照)。色の設定を変更すると、「コールグラフ」ウィンドウと「クラスグラフ」ウィンドウの両方にその設定が適用されます。

図 5-4は、「コールグラフ」ウィンドウを示しています。

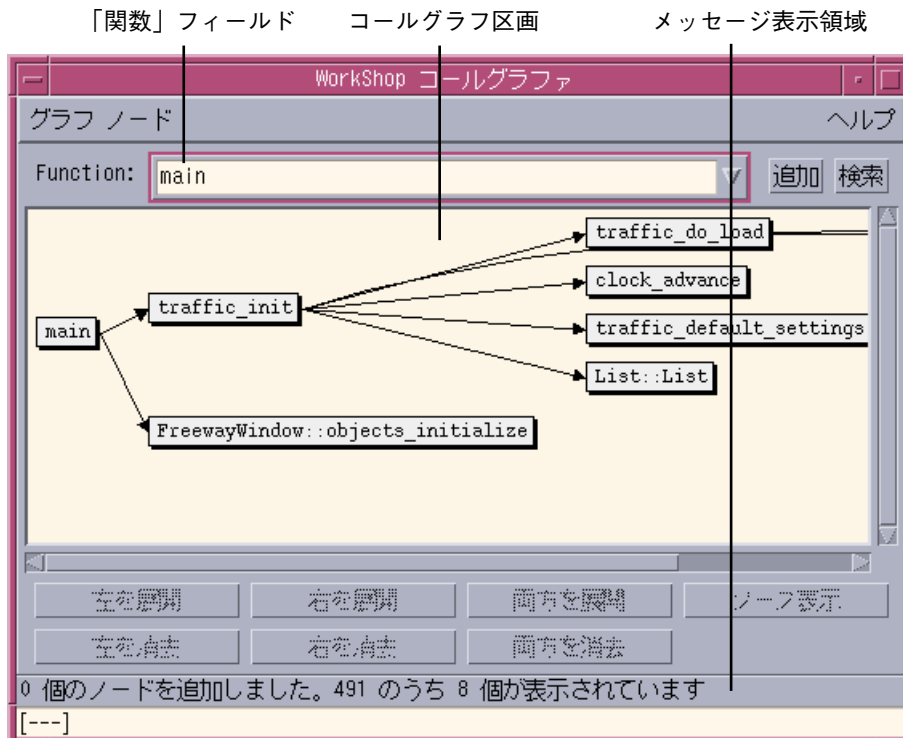


図 5-4 「コールグラフ」ウィンドウ

クラスのグラフ表示

「クラスグラフ」ウィンドウを使用して、C++ プログラムのクラスの継承構造をグラフ表示できます。具体的な使用手順とその他の情報については、オンラインヘルプを参照してください。

「クラスグラフ」ウィンドウ内のノードの背景、クラスグラフ区画の背景、ノードのボックスの枠、ノードのボックスのテキスト、ノード間の矢印に使用する色を変更する場合は、WORKSHOP リソースファイルを編集します。色の設定を変更すると、「クラスグラフ」ウィンドウと「コールグラフ」ウィンドウの両方にその設定が適用されます。色の変更については、69 ページの「「コールグラフ」ウィンドウと「クラスグラフ」ウィンドウの色」を参照してください。

図 5-5 は、「クラスグラフ」ウィンドウを示しています。



図 5-5 「クラスグラフ」ウィンドウ

クラスのブラウズ

クラスブラウザでは、次のことを行えます。

- クラスのブラウズ: クラスのリストとデータ関数のメンバーを表示します。クラスのインタフェースと相互関係を確認できます。
- クラスの関係の確認: クラスを選択して、その基底クラス、派生クラス、フレンドクラスを確認したり、現在のクラスで参照されているクラス、構造体、共用体をブラウズしたりできます。

- **クラスのグラフ表示**：「クラスブラウザ」ウィンドウで選択されているクラスのクラス階層をグラフィカルに表示できます。
- **クラスのソース表示**：エディタウィンドウに特定のクラスのソースを表示できます。

「クラスブラウザ」ウィンドウでは、クラスやそのメンバー関数、フレンド関数に関する情報を表示できます。ソースコードとライブラリ内のクラスをナビゲートすることによって、クラスがどのように定義、使用されているかを確認できます。

「クラスブラウザ」ウィンドウ (図 5-6) を開くと、現在のソースブラウザ用データベースに格納されている Class 型または Struct 型のクラスすべてを一覧表示できます。「項目」リストの右横にある 2 つのチェックボックスを使用し、表示内容 (クラスの型すべて、クラスと構造体のみ、共用体のみ) を切り替えられます。

具体的な使用手順とその他の情報については、オンラインヘルプを参照してください。

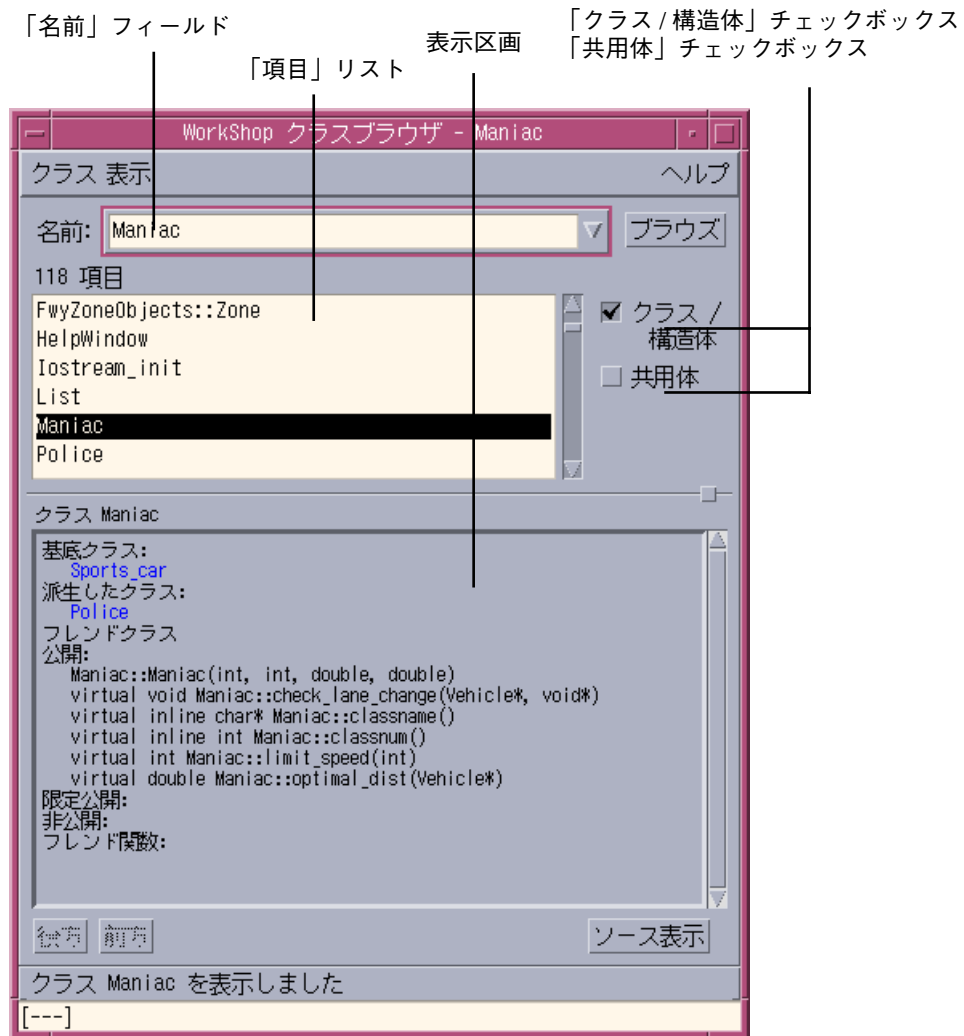


図 5-6 「クラスブラウザ」ウィンドウ

ブラウザの終了

ブラウザの処理を終了して、ブラウザ用のすべてのウィンドウを閉じるには、「ブラウザ」ウィンドウから「ブラウザ」▶「ブラウザ終了」を選択します。処理を強制終了せずに「ブラウザ」ウィンドウだけを閉じる場合は、「ブラウザ」▶「閉じる」を選択します。

第6章

プログラムのパフォーマンスの解析

この章では、標本コレクタとパフォーマンスアナライザの基本機能について説明します。UNIX の `prof` と `gprof` パフォーマンスプロファイルツールが作成する情報は、ユーザーの CPU 使用時間情報だけです。これに対し、標本コレクタとパフォーマンスアナライザでは、関数やロードオブジェクト、標本収集間隔、マルチスレッドプログラムのスレッドおよび軽量プロセス (LWP) 別に分類された広範囲の測定項目を検討できます。

- 標本コレクタは、アプリケーションの実行中にパフォーマンスデータを収集し、その収集データを実験ファイルに保存します。標本コレクタは、「デバッグ」ウィンドウの「ウィンドウ」メニューから起動します。
- パフォーマンスアナライザは、実験ファイル内のパフォーマンスデータを表示します。このパフォーマンスデータに基づいて、プログラムのパフォーマンスを解析し、改善箇所を調べることができます。パフォーマンスアナライザは、Sun WorkShop のメインウィンドウの「ツール」メニューや、「標本コレクタ」ウィンドウから起動します。

Sun WorkShop に付属するこれらのツールとその他のパフォーマンスプロファイルツールの使用方法についての詳細は、次を参照してください。

- 『プログラムのパフォーマンス解析』
- オンラインヘルプ

注 - パフォーマンスデータを収集するには、アプリケーションを構築しておく必要があります。構築については、このマニュアルの第3章とオンラインヘルプを参照してください (オンラインヘルプには、Sun WorkShop のどのウィンドウからでも「ヘルプ」メニューまたは「ヘルプ」ボタンでアクセスできます)。

パフォーマンスデータの収集

標本コレクタは、プログラムが「デバッグ」ウィンドウで実行されているときに、そのパフォーマンスデータを収集し、その収集データを実験ファイルに保存します。保存した実験ファイルは、パフォーマンスアナライザで利用できます。

表 6-1 に、収集できるデータの種類をまとめます。

表 6-1 収集できるデータの種類

種類	説明
時間ベースのプロファイルデータ	関数またはロードオブジェクトのタイミング情報
ハードウェアカウンタオーバーフローロファイルデータ	発行されたか実行された命令、キャッシュのミス、サイクル、浮動小数点演算およびその他のハードウェア演算の数
同期遅延データ	マルチスレッドプログラムおよびメッセージ転送インタフェース (MPI) プログラムにおける同期ルーチンの呼び出しに対する待ち時間
アドレス空間データ	アプリケーションのアドレス空間のページおよびセグメントの使用方法についての情報

標本コレクタは、ページフォルトや入出力データ、コンテキストの切り替え、ワークセットおよびページング統計情報などの実行時統計情報を自動的に記録します。

収集するデータの種類の選択と標本コレクタの実行についての詳細は、次を参照してください。

- 『プログラムのパフォーマンス解析』
- オンラインヘルプ

注 - 標本コレクタは、`collector` サブコマンドを使用して `dbx` から実行することもできます。または直接コマンド行から `collect` コマンドを入力して実行することもできます。詳細は、マニュアルページの `dbx(1)`、`collect(1)`、および `collector(1)` と『プログラムのパフォーマンス解析』を参照してください。

パフォーマンスデータの解析

標本コレクタでパフォーマンスデータを収集すると、パフォーマンスアナライザでそのデータを表示できます。パフォーマンスアナライザは独立したツールで、Sun WorkShop のメインウィンドウや「標本コレクタ」ウィンドウの「ツール」メニューから起動します。パフォーマンスアナライザのさまざまな表示データに基づいて、プログラムが過度に実行時間を費やしている箇所やシステム資源を効率よく利用していない箇所を特定できます。

パフォーマンスアナライザウィンドウでは、「データ」リストボックスから表示オプションを選択できます (表 6-2 を参照)。

表 6-2 表示、解析できるデータの種類の種類

オプション	説明
関数リストの表示	プログラムの関数やロードオブジェクトに関する詳細情報
概要の表示	プログラムの標本収集間隔に関する統計情報
アドレス空間の表示	プログラムのアドレス空間内のページまたはセクターの使用状況に関する情報
実行統計の表示	プログラムの実行時間全体にまたがるグローバルな統計情報

パフォーマンスアナライザでは、プログラム全体のデータを検査することも、解析対象として関数やロードオブジェクト、標本収集間隔、スレッド、LWP を個別に指定することもできます。また、パフォーマンスアナライザを使用して、リンカー用のマップファイルを作成することもできます。リンカーは、このマップファイルを使用し、プログラムがアドレス空間をより効率よく使用するようになります。

パフォーマンスアナライザの関数リストでは、一度に複数の実験を表示でき、複合データやデータの一部を選択した実験から見ることができます。

パフォーマンスアナライザについての詳細は、次を参照してください。

- 『プログラムのパフォーマンス解析』
- オンラインヘルプ

関数およびロードオブジェクトデータの表示

関数リストには、関数またはロードオブジェクトごとに次のメトリック (計測データ) に関する値 (排他的な値と包括的な値) が表示されます。

- ユーザーの CPU 使用時間や総 LWP 時間、ウォールクロック時間、システム時間、ページフォルト時間などの時間ベースのプロファイル
- ハードウェアカウンタプロファイル (使用可能な場合)
- 同期遅延データ (マルチスレッドおよび MPI プログラムの場合)

それぞれのメトリックについて、絶対値 (秒数、回数) とプログラム全体のメトリックに対する割合 (百分率) が表示できます。

関数リストに表示するメトリックや、データの並べ替えの基準にするメトリックを指定できます。また、選択した関数またはロードオブジェクトに対して指定可能な全メトリックを含むウィンドウを開くこともできます。

呼び出し元と呼び出し先のデータの表示

関数リストから「呼び出し元 - 呼び出し先」ウィンドウを開き、現在の関数とその呼び出し元、呼び出し先の排他的なデータと包括的なデータ、属性データを表示できます。現在の関数の呼び出し元または呼び出し先を選択することによって、プログラムをステップ実行することもできます (この後は、呼び出し元または呼び出し先が現在の関数になります)。表示するメトリックおよびデータソートするメトリックの種類を指定できます。詳細は、オンラインヘルプを参照してください。

注釈付きのソースコードと逆アセンブリコードの表示

プログラムのメトリックデータ付きでコンパイラのコメントでインタリーブされたソースコードおよび逆アセンブリコードを表示することによって、プログラムのパフォーマンスを 1 行ごとまたは命令ごとに調査できます。これらのメトリックデータによって、特定の関数内で最もリソースを使用している行、または最大の遅延を起している行を特定できます。コンパイラのコメントにより、コンパイラのコードの変換方法が分かります。詳細は、オンラインヘルプを参照してください。

第7章

ソースファイルのマージ

ファイルマージ機能によって、2つのテキストファイルを比較したり、2つのファイルをマージ (併合) した新しいファイルを作成したり、同じファイルの2つの編集バージョンを元のファイルと比較して、編集内容をすべて含む新しいファイルを作成したりできます。ファイルマージ機能は、2つのテキストファイルをそれぞれ読み取り専用の表示区画に読み込み、比較しやすいように横に並べて表示します。2つのファイルの相違箇所マークを付け、マージによって作成される新しいファイルを表示します。この新しいファイルを編集して、最終的なファイルを作成できます。

マージする2つのファイルを読み込むときに、3つ目のファイルとして、それら2つのファイルの元のファイル (祖先ファイル) を指定できます。祖先ファイルが指定された場合、ファイルマージ機能は、マージするファイルと祖先とを比較し、異なる行にマークを付けて、それら3つのファイルのすべてに基づいて新しいファイルを作成します。

詳細は、オンラインヘルプを参照してください (オンラインヘルプを表示するには、「ファイルマージ」ウィンドウから「ヘルプ」▶「目次」を選択します)。

ファイルの読み込み

ファイルをファイルマージに読み込むには、次の手順に従ってください。

1. Sun WorkShop メインウィンドウから「ツール」▶「ファイルマージ」を選択します。

「ファイルマージ」ウィンドウが開きます (図 7-1 を参照)。このウィンドウは、3つの区画に分かれています。同じファイルの異なるバージョンを表示する、ウィンドウ上部にある横に並べられた2つの区画と、マージ結果を表示するウィンドウ下部の区

画の3つです。上部の2つの区画は読み取り専用で、下部の区画にはファイルの2つのバージョンのいずれか一方または両方から選択された行が表示されます。下部の区画の内容を編集することによって、最終的にマージしたバージョンを作成できます。

2. 「ファイル」▶「開く」を選択します。

3. 「ディレクトリ」テキストボックスで作業用ディレクトリを選択します。

これは、ファイルの選択および保存に使用されるデフォルトのディレクトリです。このテキストボックスの右横の「...」ボタンをクリックすると、ディレクトリを選択するためのダイアログボックスが表示されます。

4. 「左のファイル」テキストボックスと「右のファイル」テキストボックスで、比較する2つのファイルを選択します。

5. ファイルを共通の祖先と比較する場合は、2つのファイルの祖先ファイルを「祖先ファイル」テキストボックスに入力します。

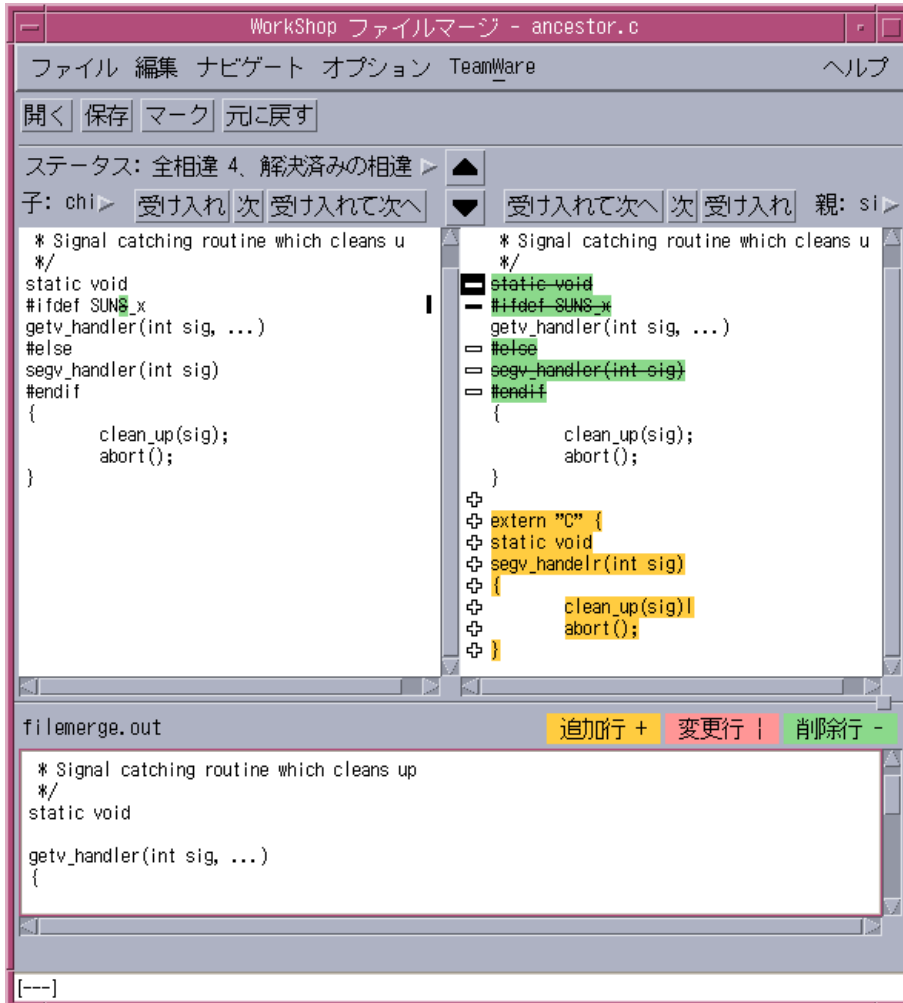
自動ファイルマージを使用するには、祖先ファイルが必要です。

6. 出力ファイルの名前を指定する場合は、「出力ファイル」テキストボックスに名前を入力します。

デフォルトでは、filemerge.out という名前のファイルが作業用ディレクトリに保存されます。

7. 「開く」をクリックして、ファイルを読み込みます。

それぞれのテキスト区画の上に左側のファイル名、右側のファイル名、出力ファイル名が表示されます。3つのファイルの比較では、ウィンドウのヘッダー部分に祖先ファイルの名前が表示されます。



相違の処理

ファイルマージは、ファイル間の相違箇所を処理します。マージする2つのファイル(または2つのうちのどちらかのファイルと祖先ファイル)の間で異なる行を検出すると、行の相違の種類に応じたグリフを使用して、2つのファイルの相違行にマークを付けます。ここでは、相違行をまとめて相違と呼びます。ファイル内の次の相違に移動すると、相違行とそのグリフが強調表示されます。

強調表示されている相違を「現在の相違」と呼びます。直前の相違を「前の相違」、直後の相違を「次の相違」と呼びます。行に対する変更が受け入れられると、相違は解決されたこととなります。残る相違は、まだ解決されていない相違です。

ファイルマージのグリフの意味

相違が簡単に見分けられるよう、ファイルマージでは、いくつかの色とグリフを使用して、相違する行を強調表示します。黄色は追加されている行、赤色は変更されている行、緑色は削除されている行を示します。

グリフの意味は、ファイルの2つのバージョンの比較(入力ファイルが2つ)か、祖先ファイルと2つのバージョンの比較(入力ファイルが3つ)かによって異なります。

入力ファイルが2つの場合

ファイルマージに2つのファイルだけが読み込まれ、それらのファイルの対応する行に相違がある場合は、それぞれのファイルの該当する行にグリフが表示されます。

- 2つの行に相違がない場合、グリフは表示されません。
- 2つの行に相違がある場合は、両方のテキスト区画のその行の先頭に縦棒(|)が表示され、相違部分が赤色で強調表示されます。
- ある行が一方のファイルにあって、もう一方にない場合は、その行が存在する方のファイルの該当する行の先頭に正符号(+)が表示され、相違部分が黄色で強調表示されます。
- 相違が解決された行には、白抜きのグリフが表示されます。

入力ファイルが 3 つの場合

マージする 2 つのファイルを読み込むときに、3 つ目のファイルとして、それら 2 つのファイルの祖先ファイルを指定できます。祖先ファイルは、マージする 2 つのファイルより古いバージョンのファイルです。祖先ファイルを指定すると、そのファイルが、マージする 2 つのファイルと比較する基準として使用されます。また、祖先ファイルを指定することによって、自動マージを行うことができます。ファイルマージは、祖先ファイルから派生したファイル、すなわち、子孫ファイル内の、祖先と異なるすべての行にマークを付け、それら 3 つのファイルに基づいてマージファイルを作成します。

マージする 2 つのファイル内の、祖先ファイルと異なる行は、変更バーのマークが付けられ、いくつかの色を使用して強調表示されます。

- 3 つのファイルの間で行に相違が存在しない場合、グリフは表示されません。
- 祖先には存在しない行が、子孫のいずれかまたは両方に存在する場合は、その行が存在するファイルの該当する行の先頭に正符号 (+) が表示され、相違部分が黄色で強調表示されます。
- 祖先に存在する行が、子孫のいずれかまたは両方の対応する行と異なる場合は、その異なる行が存在するファイルの該当する行の先頭に縦棒 (|) が表示され、相違部分が赤色で強調表示されます。
- 祖先に存在する行が、子孫のいずれかまたは両方に存在しない場合は、その行が存在しないファイルの該当する行位置の先頭に負符号 (-) が表示され、相違部分が抹消線付きの緑色で強調表示されます。
- 相違が解決された行には、白抜きグリフが表示されます。

相違間の移動

相違間を移動するには、2 つの区画の上にあるボタンか、「ナビゲート」メニューを使用します。相違を受け入れずに前の相違へ移動するには「前」ボタン、次の相違へ移動するには「次」ボタンをクリックします。特定のテキスト文字列に移動するには、「ナビゲート」▶「検索」を選択します。特定の行番号に移動するには、「ナビゲート」▶「行の移動」を選択します。

「親」と「子」の区画にあるポップアップメニューを使用して相違間を移動することもできます。どちらかの区画内でマウスの右ボタンをクリックすると、メニューが開きます。

相違の解決

相違を解決するには、左右の区画の一方で変更を受け入れます。相違を受け入れるには、次のいずれかの操作をします。

- 相違を受け入れるには、「受け入れ」ボタンをクリックします。
- 変更を受け入れて次の相違に移動するには、「受け入れて次へ」ボタンをクリックします。

詳細は、オンラインヘルプを参照してください。オンラインヘルプを表示するには、「ファイルマージ」ウィンドウから「ヘルプ」▶「目次」を選択します。

相違オプションの設定

ファイルマージがファイル間の特定の相違を無視するようにカスタマイズするには、「オプション」▶「相違オプション」を選択します。行末の空白や空白の連続、英数字の大文字と小文字の違いを無視するように設定できます。

詳細は、オンラインヘルプを参照してください(オンラインヘルプを表示するには、「ファイルマージ」ウィンドウから「ヘルプ」▶「目次」を選択します)。

ファイルの自動マージ

ファイルマージは、以下の規則に従って、相違を自動的に解決できます。

- 3つのファイルすべてについて変更のない行は、出力ファイルに追加します。
- 子孫のどちらかで行が変更されている場合は、変更後の行を出力ファイルに追加します。変更には、行全体の追加または削除、行の部分的置換があります。
- 両方の子孫で同じ行に同じ変更が加えられている場合は、その行を出力ファイルに追加します。
- 両方の子孫で同じ行が編集されていて、3つのファイルのすべてで異なる場合は、どの行も出力ファイルに追加しません。この場合は、相違の解決方法を自分で決定する必要があります。子孫の行を選択するか、マージ後の新しいファイルを手動で編集します。

ファイルマージが相違を自動的に解決すると、グリフが白抜きに変わります。ファイルマージを使用して、自動的に解決された相違が正しい選択をしているかどうかを確認できます。

「オプション」▶「自動マージ」を選択すると、自動ファイルマージ機能をオフにできます。自動ファイルマージ機能がオフの場合、出力ファイルには、3つのファイルすべてで同じ行のみが追加されるため、相違はユーザー自身が解決する必要があります。

祖先ファイルを指定しなかった場合は、ファイルマージには、2つの入力ファイル間の相違を比較する基準がありません。このため、ファイルマージは、相違のどちらの行が必要な変更内容を表しているかを判断できません。祖先を指定しないで自動ファイルマージを実行すると、自動ファイルマージをオフにした場合と同じ結果になります。すなわち、ファイルマージは、両方の入力ファイルで同じ行だけを使用してマージファイルを作成するため、相違はユーザー自身が解決する必要があります。

出力ファイルの保存

出力ファイルを保存するには、「保存」ボタンをクリックするか、「ファイル」▶「保存」を選択します。出力ファイルの名前は、「出力ファイル」テキストボックスに指定した名前です。

保存する際に出力ファイルの名前を変更するには、「別名保存」を選択し、呼び出されたウィンドウに新しいファイル名とディレクトリ名を入力します。

ファイルマージオプションの設定

「ファイルマージ」ウィンドウの「オプション」メニューを使用して、さまざまなファイルマージオプションを設定できます。このメニューから、次のことを行うことができます。

- ファイルマージ後の新しいファイルの自動作成
- ファイルの個別スクロールまたは対応する行単位の同時スクロールの制御
- 行番号および行末の表示の制御
- タブの設定

■ 空白文字と英大文字 / 小文字の区別の取り扱いの設定

詳細は、オンラインヘルプを参照してください。オンラインヘルプを表示するには、「ファイルマージ」ウィンドウから「ヘルプ」▶「目次」を選択します。

付録 A

Sun WorkShop とテキストエディタの リソース

この付録では、設定可能なリソースと、設定の変更に必要な情報を提供します。この付録は、以下の節から構成されています。

- 「リソース設定の変更」
- 「編集可能な Sun WorkShop のリソース」
- 「編集可能なテキストエディタのリソース」

リソース設定の変更

Sun WorkShop 統合プログラミング環境では、次の 2 つのリソースファイルを使用します。

- WORKSHOP: 「ブラウザ」、「デバッグ」ウィンドウなどの、Sun WorkShop のウィンドウのリソース設定が格納されています。
- ESERVE: テキストエディタのリソース設定が格納されています。

これらのリソースファイルには、2 つのバージョンがあります。1 つは CDE (共通デスクトップ環境) 用、もう 1 つは非 CDE 環境用です。CDE 版では、Motif の要素に対する汎用の色とフォントリソースが定義されていません。CDE 版では、これらの要素を CDE スタイルマネージャで制御できます。

WORKSHOP および ESERVE ファイルでは、リソースがグループ別にまとめられ、そのグループに関するコメントが含まれています。たとえば、次のグループのリソースは、テキストエディタで強調表示に使用する色を制御します。

```
! Resources for highlight colors used by WORKSHOP in the editors

WORKSHOP.curPCColor:    #8BD98B
WORKSHOP.visitPCColor:  #EDC9FF
WORKSHOP.breakptColor:  #FF9696
```

リソースファイルについての詳細は、workshop(1) のマニュアルページを参照してください。

リソースのデフォルト値を変更するには、次の手順に従ってください。

1. 変更するリソースに従い、ホームディレクトリ、または環境変数の XFILESEARCHPATH か XAPPLRESDIR で指定したディレクトリに、WORKSHOP または ESERVE というファイルを作成します。
2. 編集するリソースファイルが存在するディレクトリに移動します。
リソースファイルは、次のシステムやネットワーク上の Sun WorkShop インストールディレクトリに置かれます。

```
/opt/SUNWspro/WS6/lib/locale/lang/app-defaults/CDE
```

```
/opt/SunWspro/WS6/lib/locale/lang/app-defaults/non-CDE
```

lang は、使用しているロケール名 (c や ja など) です。

注 – Sun WorkShop 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、システム管理者に確認してください。

3. WORKSHOP または ESERVE ファイルから、変更するリソースとそのデフォルト値をコピーします。
4. 自分のホームディレクトリに作成したリソースファイルに、コピーしたリソースとデフォルト値をペーストします。
5. この付録の指示に従ってリソース値を変更します。
6. ファイルを保存します。

7. Sun WorkShop 統合プログラミング環境を起動 (または、Sun WorkShop をいったん終了して再起動) します。

編集可能な Sun WorkShop のリソース

WORKSHOP リソースファイルで変更可能な Sun WorkShop のリソースを以下に示します (方法については、65 ページの「リソース設定の変更」を参照してください)。

- 68 ページの「エディタウィンドウの強調色」
- 68 ページの「[「データグラフ」ウィンドウの色」
- 69 ページの「[「コールグラフ」ウィンドウと「クラスグラフ」ウィンドウの色」
- 69 ページの「警告音」
- 70 ページの「デバッグボタン」
- 70 ページの「[「dbx コマンド」ウィンドウと「プログラム入出力」ウィンドウの出力行」
- 71 ページの「プロジェクト構築コマンド」
- 71 ページの「[「Web での最新情報」の表示に使用するブラウザ」
- 71 ページの「ハイパーリンクウィンドウの文字フォント」
- 73 ページの「ハイパーリンク用リソース」
- 74 ページの「テキストの自動改行」
- 74 ページの「垂直スクロールバー」
- 74 ページの「Motif に固有のリソース」
- 76 ページの「ウィンドウの前景色と背景色」
- 77 ページの「スクロールバーの背景とトグルボタンの色」

Sun WorkShop TeamWare コンポーネントおよびメインウィンドウの「ツール」メニューから起動するコンポーネントが、Sun WorkShop 統合プログラミング環境本体のコンポーネントに対するリソース設定の影響を受けることはありません。

注 – デフォルトの色を変更して、指定されている以外の色を使用すると、Sun WorkShop 統合プログラミング環境によってカラーマップが使い切られることがあります。

エディタウィンドウの強調色

表 A-1 のリソースは、テキストエディタウィンドウで、関数、ブレイクポイント、問い合わせに一致する文字列、ソースコードの構築エラーを強調表示するときに使用される色を制御します。

表 A-1 エディタウィンドウで使用される強調色のリソース

リソース名	内容	デフォルト値
WORKSHOP.curPCColor:	現在の関数	#8BD98B
WORKSHOP.visitPCColor:	表示関数	#EDC9FF
WORKSHOP.breakptColor:	ブレイクポイント	#FF9696
WORKSHOP.disabledBreakptColor	休止中のブレイクポイント	#BDBDBD
WORKSHOP.matchColor:	パターンまたはシンボルによる検索に一致する文字列	#99CFFF
WORKSHOP.errorColor:	現在の構築エラー	#FFCC40

「データグラフ」ウィンドウの色

表 A-2 のリソースは、デバッガの「データグラフ」ウィンドウのグラフで使用される色を制御します。『dbx コマンドによるデバッグ』を参照してください。

表 A-2 「データグラフ」ウィンドウの色のリソース

リソース名	内容	デフォルト値
WORKSHOP.dgLineColor:	線グラフの色	#0000FF
WORKSHOP.dgFillColor:	塗りつぶしグラフの色	#FDF5E6
WORKSHOP.dgMeshColor:	メッシュグラフの色	#0000FF

「コールグラフ」ウィンドウと「クラスグラフ」ウィンドウの色

表 A-3 のリソースは、ノード、ノード間を結ぶ線 (または矢印)、「コールグラフ」ウィンドウ (図 5-4) と「クラスグラフ」ウィンドウ (図 5-5) のグラフ区画の背景色を制御します。

表 A-3 「コールグラフ」と「クラスグラフ」のウィンドウで使用される色のリソース

リソース名	内容	デフォルト値
WORKSHOP*labelNodeBackground:	各ノードの背景色	#EFEFEF
WORKSHOP*viewBackground:	グラフ区画の背景色 (デフォルトは X の OldLace)	#FDF5E6
非強調時のノードの属性		
WORKSHOP*arcForeground:	ノード間の矢印	#000000
WORKSHOP*nodeForegroundColor:	ノードの境界線	#000000
WORKSHOP*labelNodeForeground:	ノードのテキスト	#000000
強調時のノードの属性		
WORKSHOP*arcHighlightColor:	ノード間の矢印	#FF0000
WORKSHOP*nodeHighlightColor:	ノードの境界線	#FF0000

警告音

表 A-4 のリソースでは、警告音を有効または無効に設定できます。使用可能な値は、-XmBell と -XmNONE です。

表 A-4 警告音の有効 / 無効を切り替えるリソース

リソース名	内容	デフォルト値
WORKSHOP*audibleWarning	警告音を有効または無効にする	XmBell

デバッグボタン

表 A-5 のリソースでは、dbx を起動した際に、デバッグとテキストエディタのボタンが無効になるまでの時間をミリ秒単位で設定できます。デバッガとテキストエディタのボタンが無効にすると、コードをステップ実行している場合に、ボタンが点滅しなくなります。処理速度が遅いシステムや ISDN 回線を介して Sun WorkShop ツールを実行している場合は、この値を大きくした方がよい場合があります。

表 A-5 デバッガのボタンが無効になるまでの遅延時間を指定するリソース

リソース名	内容	デフォルト値
WORKSHOP.ButtonDisableDelay	dbx の起動時にデバッグとテキストエディタのボタンが無効になるのを遅らせる	250

「dbx コマンド」ウィンドウと「プログラム入出力」ウィンドウの出力行

表 A-6 のリソースでは、「dbx コマンド」ウィンドウと「プログラム入出力」ウィンドウに保存される出力行数を設定します。

表 A-6 「dbx コマンド」ウィンドウおよび「プログラム入出力」ウィンドウに保存される出力行数を設定するリソース

リソース名	デフォルト値
WORKSHOP*dtTerm.saveLines	1000

プロジェクト構築コマンド

表 A-7 に示すリソースで、プロジェクトの構築に使用されるプロジェクト構築 (make) コマンドを設定できます。プロジェクト構築コマンドは、`-f` フラグの他に、ターゲットおよびマクロオペランドを受け付ける必要があります。たとえば、特定の警告を除外して、構築コマンドにフラグを渡すラッパーを独自に作成できます。詳細は、`dmake(1)` および `make(1)` のマニュアルページを参照してください。

表 A-7 プロジェクト構築コマンドのリソース

リソース名	デフォルト値
<code>WORKSHOP.ProjectMakeCommand</code>	<code>dmake -m serial</code>

「Web での最新情報」の表示に使用するブラウザ

表 A-8 のリソースでは、Sun WorkShop の「Web での最新情報」ページの表示に使用するブラウザのデフォルトのパスを変更できます。「Web での最新情報」ページを表示するには、任意のウィンドウから「ヘルプ」▶「Web での最新情報」を選択します。

表 A-8 「Web での最新情報」の表示に使用するブラウザのパスを変更するリソース

リソース名	内容	デフォルト値
<code>WORKSHOP.browser</code>	「Web での最新情報」の表示に使用するブラウザのパス	<code>netscape</code>

ハイパーリンクウィンドウの文字フォント

Sun WorkShop のウィンドウでは、別のウィンドウへのハイパーリンクを使用して、その関連情報を表示できます。たとえば、「構築」ウィンドウの構築エラーをクリックすると、そのエラーを含むソースコードがエディタウィンドウに表示されます。リソースには、ハイパーリンク表示で使用する非 ASCII 文字を複数バイト文字と解釈するように指示するフラグの機能を持つものがあります。複数バイト文字は、リソースで設定されたフォントで表示されます。こうしたリソースは、非 ASCII 文字を複数バイト文字と解釈するロケールでのみ設定してください。

WORKSHOP リソースファイルで使用される文字フォントのリソース名は、次のとおりです。

```
WORKSHOP*HTML*WCfont :  
WORKSHOP*HTML*boldWCFont :  
WORKSHOP*HTML*plainWCFont :  
WORKSHOP*HTML*plainboldWCFont :  
WORKSHOP*HTML*Font :  
WORKSHOP*HTML*boldFont :  
WORKSHOP*HTML*plainFont :  
WORKSHOP*HTML*plainboldFont :
```

各ワイド文字 (wc) フォントのリソースは、非 wc フォントのリソースに対応しています。wc フォントのリソースを設定すると、設定されたサイズによって、wc フォントや対応する非 wc フォントで書かれたテキスト要素の行間隔と基準線が決定されます。これによって、ASCII 文字と複数バイト文字が混在する場合でも、行の間隔が一定になります。また、非 wc フォントのみで書かれる行の書式にも wc フォントのサイズが使用されます。

複数バイト文字のハイパーリンク表示に wc フォントが設定されていて、wc フォントのリソースを変更した場合、wc フォントのサイズと間隔は、非 wc フォントのサイズや間隔と比例したものになります。このため、プロポーショナルな書式にするには、非 wc フォントのリソースを変更しなければならないことがあります。

ハイパーリンク用リソース

表 A-9 のリソースでは、英語版 Sun WorkShop のウィンドウとダイアログボックスのハイパーリンクで使用されるフォントの種類、太さ、アングルを設定します。Sun WorkShop ウィンドウのハイパーリンクの例については、「構築」ウィンドウの構築エラーリンクを示している図 3-4 を参照してください。

表 A-9 ハイパーリンクフォントのリソース

リソース名	デフォルト値
WORKSHOP*HTML*boldFont	--gothic-bold-r-normal--14-120-75-75-m-60-jisx0201.1976-0
WORKSHOP*HTML*plainFont	--gothic-medium-r-normal--14-120-75-75-m-60-jisx0201.1976-0
WORKSHOP*HTML*plainboldFont	--gothic-medium-r-normal--14-120-75-75-m-60-jisx0201.1976-0

表 A-10 に、複数バイト文字のあるロケール用のハイパーリンク WC フォントを示します。設定すると、HTML 表示に書かれる非 ASCII 文字は複数バイト文字として解釈され、リソースで設定したフォントで表示されます。

表 A-10 複数バイト文字用のハイパーリンクフォントのリソース

リソース名	デフォルト値
WORKSHOP*HTML*WCfont	--gothic-medium-r-normal--14-120-75-75-m-120-jisx0208.1983-0
WORKSHOP*HTML*italicWCFont	--gothic-medium-r-normal--14-120-75-75-m-120-jisx0208.1983-0
WORKSHOP*HTML*boldWCFont	--gothic-bold-r-normal--14-120-75-75-m-120-jisx0208.1983-0
WORKSHOP*HTML*fixedWCFont	--gothic-medium-r-normal--14-120-75-75-m-120-jisx0208.1983-0
WORKSHOP*HTML*fixedboldWCFont	--gothic-bold-r-normal--14-120-75-75-m-120-jisx0208.1983-0
WORKSHOP*HTML*fixeditalicWCFont	--gothic-medium-r-normal--14-120-75-75-m-120-jisx0208.1983-0

テキストの自動改行

表 A-11 のリソースでは、Sun WorkShop ウィンドウでテキストを自動改行するか、新しい行を開始するかを設定できます。デフォルト値は true で、この場合、ウィンドウの端でテキストが自動改行されます。

表 A-11 テキストの自動改行を設定するリソース

リソース名	デフォルト値
WORKSHOP*HTML*wrapPreformatText	True

垂直スクロールバー

表 A-12 のリソースでは、垂直スクロールバーの有効/無効を切り替えることができます。

表 A-12 垂直スクロールバーの有効 / 無効を切り替えるリソース

リソース名	デフォルト値
WORKSHOP*HTML*verticalScrollbarAlways	True

Motif に固有のリソース

表 A-13 から表 A-17 に、Motif 環境のみに使用され、CDE では使用されないリソースを示します。

表 A-13 Motif (非 CDE) 固有のフォントのリソース

リソース名	内容	デフォルト値
labelFontList	ラベルの フォント	--gothic-medium-r-normal--14--*-*m-*-*-*:
buttonFontList	ボタンの フォント	--gothic-medium-r-normal--14--*-*m-*-*-*:
textFontList	リストの フォント	--gothic-medium-r-normal--14--*-*m-*-*-*:

リソースファイルには、表 A-14 に示すような、個々の Sun WorkShop ウィンドウに対するフォントのリソース設定も含まれています。特定の Sun WorkShop ウィンドウのフォントを変更する場合は、リソース設定行のコメント指定を解除してください。

表 A-14 個々のウィンドウのフォントのリソース

リソース名	デフォルト値
WORKSHOP*ipeDbxCommandWindow*userFont	--lucidatypewriter-medium-r-normal--12--*--*--* --iso8859-1
WORKSHOP*ipeProgramIOShell*userFont	--lucidatypewriter-medium-r-normal--12--*--*--* --iso8859-1
WORKSHOP*threadsList*fontList	--lucidatypewriter-medium-r-normal--12--*--*--* --iso8859-1
WORKSHOP*handlerList*fontList	--lucidatypewriter-medium-r-normal--12--*--*--* --iso8859-1
WORKSHOP*processList*fontList	--lucidatypewriter-medium-r-normal--12--*--*--* --iso8859-1

表 A-15 のリソースは、表などのテーブル形式のテキストに適用されます。

表 A-15 テーブル形式のウィンドウで使用されるフォントのリソース

リソース名	デフォルト値
WORKSHOP*DataMonospacedFont	--gothic-medium-r-normal--14--*--*--m--*--*--*:

ウィンドウの前景色と背景色

表 A-16 のリソースでは、大部分の Sun WorkShop ウィンドウで使用される前景色と背景色を制御します。

表 A-16 Sun WorkShop のウィンドウ、ダイアログボックス、メニュー、ボタンで使用される色のリソース

リソース名	内容	デフォルト値
WORKSHOP*foreground	ウィンドウの前景色 (ラベルなどのテキスト)	#000000
WORKSHOP*XmTextField*background	テキストボックスの背景色	#FFFFFF
WORKSHOP*XmText*background	テキストの色	#FFFFFF
WORKSHOP*threadsList.background	スレッド区画の背景色	#FFFFFF
WORKSHOP*ipeDbxCommandWindow*dtTerm.background	「dbx コマンド」 ウィンドウの背景色	#FFFFFF
WORKSHOP*ipeProgramIOShell*dtTerm.background	「プログラム入出力」ウィ ンドウの背景色	#FFFFFF
WORKSHOP*XmDrawingArea.background	スタック区画やデータ表示 などの背景色	#FFFFFF
WORKSHOP*background	Sun WorkShop ウィンドウ の背景色	#DEDEDE
WORKSHOP*XmPushButton*background	ボタンの背景色	#DEDEDE
WORKSHOP*XmMenuShell*background	メニューの背景色	#DEDEDE
WORKSHOP*XmList*background	「ブラウズ」ウィンドウの 「一致」リストなど、リス トの背景色	#DEDEDE
WORKSHOP*topShadowColor	ボタンやテキストボックス などの上部および左端の影 の色	#FFFFFF

スクロールバーの背景とトグルボタンの色

表 A-17 に、スクロールバーの背景に使用する色とトグルのオン・オフを表すトグルボタンの色のリソースを示します。

表 A-17 スクロールバーの背景やトグルボタンの色のリソース

リソース名	内容	デフォルト値
WORKSHOP*HTML*troughColor	スクロールバーの背景色	#DEDEDE
WORKSHOP*XmToggleButton.selectColor	選択された状態のチェックボックスの色	#FF9696
WORKSHOP*XmToggleButton.fillOnSelect	選択されたときチェックボックスを塗り潰す	true
WORKSHOP*XmToggleButtonGadget.selectColor	選択された状態のラジオボタンの色	#FF9696
WORKSHOP*XmToggleButtonGadget.fillOnSelect	選択されたときラジオボタンを塗り潰す	true

編集可能なテキストエディタのリソース

RESERVE リソースファイルで変更可能な Sun WorkShop のリソースを以下に示します (方法については、65 ページの「リソース設定の変更」を参照してください)。

- 78 ページの「テキストエディタのデフォルトパス名」
- 78 ページの「点滅ポインタ」
- 79 ページの「Motif 環境のテキストエディタのフォント」
- 79 ページの「テキストエディタウィンドウの色」
- 79 ページの「スクロールリストの背景色」
- 80 ページの「書き込み可能領域の背景色」
- 80 ページの「式のバルーン評価ポップアップウィンドウの寸法」
- 80 ページの「警告音」

テキストエディタのデフォルトパス名

表 A-18 に、指定されたテキストエディタを編集サーバーが呼び出すために使用するリソースを示します。絶対パスが指定されている場合は、そのパスのエディタが実行されます。

表 A-18 テキストエディタのデフォルトパスを設定するリソース

リソース名	デフォルト値
ESERVE*defaultGnuEmacsPath	mule
ESERVE*defaultXEmacsPath	xemacs-mule
ESERVE*defaultNEditPath	nedit
ESERVE*defaultGVimPath	gvim

表 A-18 のリソース値は、絶対パスでもコマンドのベース名 (myfavoriteemacs など) でもかまいません。

ベース名が使用された場合は、PATH 環境変数から呼び出されます。

点滅ポインタ

表 A-19 に、テキストエディタウィンドウのポインタを点滅しないように変更するリソースを示します。ポインタは、デフォルトでは点滅するように設定されています。点滅しないようにするには、0 に設定します。

表 A-19 点滅ポインタのリソース

リソース名	デフォルト値
ESERVE*DtTerm.blinkRate	250

Motif 環境のテキストエディタのフォント

表 A-20 に、Motif 環境のみに使用され、CDE では使用されないテキストエディタウィンドウのフォントリソースを示します。

表 A-20 Motif (非 CDE) 固有のフォントのリソース

リソース名	デフォルト値
labelFontList	--gothic-medium-r-normal--14----*--m--*--*--*
buttonFontList	--gothic-medium-r-normal--14----*--*--m--*--*--*
textFontList	--gothic-medium-r-normal--14----*--*--m--*--*--*

テキストエディタウィンドウの色

表 A-21 に、テキストエディタウィンドウの前景色と背景色のリソースを示します。

表 A-21 テキストエディタのウィンドウ、ダイアログボックス、メニュー、ボタンに使用する色のリソース

リソース名	内容	デフォルト値
ESERVE*foreground	スクロールバーの前景色 (ラベルなどのテキスト)	black (黒)
ESERVE*background	ウィンドウの背景色	#dededededede
ESERVE*XmPushButton*background	ボタンの背景色	#dededededede
ESERVE*XmMenuShell*background	メニューの背景色	#dededededede

スクロールリストの背景色

表 A-22 に、テキストエディタウィンドウで使用される、スクロールリストの背景色を設定するリソースを示します。

表 A-22 スクロールリストの背景色を設定するリソース

リソース名	内容	デフォルト値
ESERVE*XmList*background	スクロールリストの背景色	#dededededede

書き込み可能領域の背景色

表 A-23 に、テキストエディタウィンドウ内のテキスト領域の色を設定するリソースを示します (Emacs、XEmacs には適用されません)。

表 A-23 書き込み可能なテキスト領域の色を設定するリソース

リソース名	デフォルト値
ESERVE*XmTextField*background	white (白)
ESERVE*XmText*background	white (白)
ESERVE*dtTerm*background	white (白)
ESERVE*readwriteBackground	white (白)

式のバルーン評価ポップアップウィンドウの寸法

表 A-24 に示すリソースには、テキストエディタウィンドウ内のカーソル位置にある式の現在値をすぐに表示する、バルーン評価ポップアップウィンドウの最大サイズを設定します。幅の単位は文字数、高さの単位は行数です。

表 A-24 バルーン評価ポップアップウィンドウのサイズ

リソース名	デフォルト値
ESERVE.balloonWidth	60
ESERVE.balloonHeight	20

警告音

表 A-25 のリソースでは、テキストエディタウィンドウでの警告音をオフにできます。使用可能な値は、-XmBell と -XmNONE です。

表 A-25 警告音のオン / オフを切り替えるリソース

リソース名	内容	デフォルト値
ESERVE*audibleWarning	警告音をオンまたはオフにする	XmBell

付録 B

make ユーティリティとメイクファイル

make ユーティリティとメイクファイルは、Sun WorkShop 統合プログラミング環境でのアプリケーションの自動構築に役立ちます。この付録では、make ユーティリティ、メイクファイル、メイクファイルマクロに関する基本情報を提供します。また、メイクファイルオプションを設定したり、メイクファイルマクロを追加、変更、無効にするためのダイアログボックスについても説明します。独自のメイクファイルを作成しないでプログラムを構築する場合は、18 ページの「プログラムの構築」と 19 ページの「デフォルト値を使用した構築」を参照してください。

make ユーティリティを使用して、プログラムをコンパイルし、リンクする方法を指示できます。一般に、大きなアプリケーションは、複数のソースファイルおよびインクルードファイルの集まりであり、複数のライブラリとリンクする必要があります。ソースファイルの一部を変更すると、その部分を再コンパイルし、再リンクする必要があります。このプロセスを、アプリケーションを構成するファイル間の相互関係と、それぞれの部分の再コンパイルと再リンクに必要なコマンドを指定することによって自動化できます。make は、メイクファイル内の指定に基づいて、再コンパイルが必要なファイルだけ再コンパイルし、指定されたオプションとライブラリがリンクされるようにします。

プログラム開発ツールとして make を使用方法については、『make 改訂版』(発行: 株式会社オライリー・ジャパン) などの市販本を参照してください。

メイクファイル

makefile と呼ばれるファイルは、どのような構造でソースファイルやオブジェクトファイルが他のファイルに依存しているかを make ユーティリティに示します。また、それらファイルのコンパイルやリンクに必要なコマンドも、このファイルで定義されます。

構築する各ファイル (実行のステップ) をターゲットと呼びます。メイクファイルの各エントリは、ターゲットオブジェクトの依存関係とそのオブジェクトを構築 (make) するために必要なコマンドを表す規則です。メイクファイル内の規則の構造は次のとおりです。

```
target: dependencies-list
TAB   build-commands
```

target はターゲット、*dependencies-list* は依存関係リスト、*build-commands* は構築コマンドです。依存関係リストの各エントリは、ターゲットファイル名を示す行で始まり、その後にそのターゲットが依存するすべてのファイルが続きます。構築コマンドのエントリは、ターゲットファイルを構築する Bourne シェルコマンドを指定した 1 行以上の行で構成されます。これらの行はそれぞれ、タブでインデントする必要があります。

FORTRAN 77 の例

次のようなソースファイルとメイクファイルから構成されるプログラムがあると仮定します。

- makefile
- commonblock
- computepts.f
- pattern.f
- startupcore.f

pattern.f と computepts.f は共に、commonblock をインクルードします。ここでは、各 .f ファイルをコンパイルし、これら 3 つの再配置可能ファイルを、一連のライブラリとリンクし、pattern という名前のプログラムに作成します。

この場合のメイクファイルは次のようになります。

コード例 B-1 Fortran 77 用のメイクファイル例

```
pattern: pattern.o computepts.o startupcore.o
        f77 pattern.o computepts.o startupcore.o -lcore77 \
        -lcore -lsunwindow -lpixrect -o pattern
pattern.o: pattern.f commonblock
        f77 -c -u pattern.f
computepts.o: computepts.f commonblock
        f77 -c -u computepts.f
startupcore.o: startupcore.f
        f77 -c -u startupcore.f
```

このメイクファイルの先頭行は、`pattern` の作成が `pattern.o`、`computepts.o`、`startupcore.o` に依存していることを表します。2 行目以降は、再配置可能な `.o` ファイルとライブラリから `pattern` を作成するためのコマンドです。

C++ の例

次のようなソースファイルとメイクファイルから構成されるプログラムがあると仮定します。

- `manythreads.cc`
- `Makefilemany.cc`
- `thr.cc`
- `misc.h`
- `defines.h`

ターゲットファイルは、次のとおりです。

- `many`
- `manythreads`
- `thrI`

この場合のメイクファイルは次のようになります。

コード例 B-2 C++ 用のメークファイル例

```
all: many manythreads thrI
many: many.cc
    CC -o many many.cc -g -D_REENTRANT -lm -lnsl -lsocket -lthread
thrI: thr.cc
    CC -o thrI thr.cc -g -D_REENTRANT -lm -lnsl -lsocket -lthread
manythreads: manythreads.cc
    CC -o manythreads -g -D_REENTRANT manythreads.cc -lnsl \
    -lsocket -lthread
```

このメークファイルの先頭行では、all というラベルを使用して、一連のターゲットファイルをまとめています。2 行目以降は、ソースファイルのいずれかと依存関係にある 3 つのターゲットファイルを作成するためのコマンドです。

make ユーティリティ

make ユーティリティを起動するには、コマンド行から次のコマンドを入力します。

```
% make
```

「構築オプション」ダイアログボックスを使用して、アプリケーションに応じて make コマンドに必要なオプションを指定できます。20 ページの「構築オプションの指定」を参照してください。

make ユーティリティは、現在のディレクトリ内にある makefile または Makefile という名前のファイルを検索し、そのファイルから指示を受け取ります。

make ユーティリティは、次の処理を行います。

1. メークファイルを読み取り、処理すべきすべてのターゲットファイル、ターゲットファイルが依存するファイル、ターゲットファイルの構築に必要なコマンドを特定する。
2. それぞれのファイルが最後に変更された日時を検索する。
3. 依存するファイルよりも古いターゲットファイルがあれば、そのターゲットファイルに対してメークファイルで指定されているコマンドを使用して再構築する。

メイクファイルを簡単に作成できるように、make ユーティリティには、ターゲットファイルの接尾辞に応じたデフォルトの規則があります。ファイルの接尾辞が .f の場合は、make は f77 コンパイラを使用し、FFLAGS マクロで指定されたすべてのフラグ、-c フラグ、コンパイルするソースファイル名を引数として渡します。

次の例では、この規則を 2 回利用しています。

コード例 B-3 make のデフォルトの接尾辞規則

```
OBJ = pattern.o computepts.o startupcore.o
FFLAGS=-u
pattern: $(OBJ)
    f77 $(OBJ) -lcore77 -lcore -lsunwindow \
    -lpixrect -o pattern
pattern.o: pattern.f commonblock
    f77 $(FFLAGS) -c pattern.f
computepts.o: computepts.f commonblock
startupcore.o: startupcore.f
```

make はデフォルトの規則を使用して、computepts.f および startupcore.f をコンパイルします。同様に、ファイルの接尾辞が .f95 の場合は、f95 コンパイラが使用されます。

マクロ

make ユーティリティのマクロ機能を使用すると、パラメータを使用しないで、簡単に文字列を置き換えることができます。たとえば、ターゲットプログラム pattern を構成する再配置可能なファイルの一覧を 1 つのマクロ文字列として表すと、変更する際も処理が簡単です。make のマクロについては、make(1S) のマニュアルページを参照してください。

マクロ文字列は、次の形式で定義します。

```
% make NAME=string
```

マクロ文字列であることを示すには、\$(NAME) の形式を使用します。この形式が使用されている場合、make ユーティリティは、自動的にそのマクロ文字列を実際の値に置き換えます。

次に示す例では、すべてのオブジェクトファイルを指定したマクロ定義をメイクファイルの先頭に追加します。

```
OBJ=pattern.o computepts.o startupcore.o
```

これで、ターゲット `pattern` に対して、メイクファイル内で依存関係のリストと `f77` リンクコマンドの両方にこのマクロを使用できます。

```
pattern: $(OBJ)
    f77 $(OBJ) -lcore77 -lcore -lsunwindow \
    -lpixrect -o pattern
```

マクロ文字列の名前が1文字の場合は、括弧を省略できます。

「マクロ作成」ダイアログボックスを使用すると、Sun WorkShop のターゲットの「マクロ」リストにマクロを追加したり、リストからマクロを削除したり、リスト中のメイクファイルマクロに値を割り当て直すことができます。このダイアログボックスの詳細については、21 ページの「メイクファイルマクロの使用法」を参照してください。

`make` のコマンド行オプションを使用して、メイクファイルマクロの初期値を無効にできます。たとえば、メイクファイルの先頭に以下の行を指定します。

```
FFLAGS=-u
```

次に、`computepts.f` のコンパイル行を以下のようにします。

```
f77 $(FFLAGS) -c computepts.f
```

最終的なリンクを以下のようにします。

```
f77 $(FFLAGS) $(OBJ) -lcore77 -lcore -lsunwindow \
    lpixrect -o pattern
```

引数を指定しないで make コマンドだけを実行すると、上記で設定した FFLAGS の値が利用されます。また、次のようにコマンド行で指定すると、値が上書きされて優先されます。

```
% make "FFLAGS=-u -O"
```

make コマンド行の FFLAGS マクロの定義はメイクファイルの初期設定よりも優先され、-o フラグと -u フラグの両方が f77 に渡されます。コマンド行で "FFLAGS=" と指定し、マクロをリセットすることもできます。

付録 C

dmake ユーティリティの使用

この付録では、分散メーク (dmake) ユーティリティで構築プロセスを複数のホストに分散し、複数のワークステーションやマルチプロセッサ上で並列的にプログラムを構築する方法について説明します。dmake(1) のマニュアルページも参照してください。

基本概念

分散メーク (dmake) は、make ユーティリティのスーパーセットです。dmake を使用することにより、多数のプログラムで構成される大規模なプロジェクトの構築プロセスを複数のワークステーション (マルチプロセッサシステムの場合は複数の CPU) に分散して、並列的に作業を進めることができます。

dmake ホスト上で dmake を実行すると、構築サーバーにジョブが分散されます。ジョブは dmake ホストにも割り当てることができ、その場合は、dmake ホストも構築サーバーと見なされます。dmake ユーティリティは、構築が並列可能であると判断した、メークファイル内のターゲットに基づいてジョブを分散します。dmake ホストから、使用する構築サーバーや各構築サーバーに割り当てる dmake ジョブ数を制御できます。また、構築サーバーで実行可能な dmake ジョブ数を制限することもできます。

dmake ジョブの分散は、次の 2 通りの方法で制御されます。

1. 構築サーバーとして使用するマシンと各構築サーバーに分散するジョブ数: dmake ユーザーが dmake ホストからを指定します。
2. サーバーに割り当て可能な総 dmake ジョブ数: 構築サーバーの所有者 (構築サーバー構成ファイル /etc/opt/SPROdmake/dmake.conf を編集する権限を持つユーザー) が制御できます。

注 - 「構築」ウィンドウから `dmake` を使用する際の構築サーバーとジョブの指定方法については、オンラインヘルプを参照してください。コマンド行から `dmake` にアクセスする場合は、`dmake(1)` のマニュアルページを参照してください。

`dmake` を理解するには、次のことを理解する必要があります。

- `dmake` ホスト
- 構築サーバー

dmake ホスト

`dmake` ホストとは、`dmake` コマンドの発行元のマシンです。`dmake` ユーティリティは、実行時構成ファイルの内容に基づいて、ジョブをどこに分散するかを決定します。一般に、このファイルは、`dmake` ホストのホームディレクトリ内に、`.dmakerc` という名前で置かれている必要があります。`dmake` ユーティリティは、以下の順序で実行時構成ファイルを探します。

1. コマンド行で `-c` オプションを使用して指定されたパス名
2. `DMAKE_RCFILE` という名のメイクファイルマクロを使用して指定されたパス名
3. `DMAKE_RCFILE` 環境変数を使用して指定されたパス名
4. `$(HOME)/.dmakerc`

実行時構成ファイルが見つからない場合、`dmake` ユーティリティは `dmake` ホストに 2 つのジョブを割り当てます。

実行時構成ファイルによって、各構築サーバーのリストと各構築サーバーに分散するジョブの数を指定できます。次に、`.dmakerc` ファイルの例を示します。

コード例 C-1 `.dmakerc` ファイル

```
# 自分のマシン。この行により dmake は自分のマシンへジョブを分散する。
falcon { jobs = 1 }
hawk
eagle { jobs = 3 }
# 上司のマシン。たいてい、会議中。
heron { jobs = 4 }
avocet
```

falcon、hawk、eagle、heron、avocet は構築サーバー名です。これらの構築サーバーのそれぞれに割り当てるジョブ数を指定できます。デフォルトのジョブ数は2つです。# から始まる行はすべてコメントと解釈されます。上記の例の構築サーバーのリストには、dmake ホストでもある falcon が指定されています。dmake ホストを構築サーバーとして指定することもできます。実行時構成ファイルに指定されていないホストに、dmake ジョブが割り当てられることはありません。

実行時構成ファイルで、複数の構築サーバーグループを定義できます。これにより、状況に応じて構築サーバーグループを柔軟に切り換えることができます。たとえば、異なるオペレーティングシステムでの構築処理ごとに、複数の異なる構築サーバーグループを定義できます。また、特殊なソフトウェアがインストールされた複数の異なる構築サーバーグループを定義することもできます。

次に、構築サーバーのグループを定義した実行時構成ファイルの例を示します。

コード例 C-2 構築サーバーグループを定義した .dmakerc ファイル

```
earth  { jobs = 2 }
mars   { jobs = 3 }

group lab1 {
    host falcon { jobs = 3 }
    host hawk
    host eagle  { jobs = 3 }
}

group lab2 {
    host heron
    host avocet { jobs = 3 }
    host stilt  { jobs = 2 }
}

group labs {
    group lab1
    group lab2
}

group sunos5.x {
    group labs
    host jupiter
    host venus  { jobs = 2 }
    host pluto  { jobs = 3 }
}
```

構築サーバーのグループは `group` キーワードを使用して指定し、そのメンバーは中括弧 (`{}`) で区切ります。グループのメンバーとなる構築サーバーは、オプションの `host` キーワードを使用して指定します。グループを他のグループのメンバーにすることができます。個々の構築サーバーを、構築サーバーのグループを含んでいる実行時構成ファイルに指定することもできます。この場合、構築サーバーは無名のグループのメンバーとみなされます。

`dmake` ユーティリティは、次の順序で各ジョブを分散させます。

1. コマンド行で `-g` オプションを使用して指定されたグループ
2. `DMAKE_GROUP` という名のメイクファイルマクロで指定されたグループ
3. `DMAKE_GROUP` 環境変数を使用して指定されたグループ
4. 実行時構成ファイルで指定された最初のグループ

`dmake` ユーティリティを使用すると、構築サーバーごとに異なる実行パスを指定できます。デフォルトでは、`dmake` は、論理サーバー内の `dmake` ホストと同じ論理パス上にある `dmake` サポートバイナリを検索します。構築サーバーの代替パスは、たとえば次のように、ホスト属性として `.dmakerc` ファイルに指定できます。

コード例 C-3 構築サーバーの代替パスを指定した `.dmakerc` ファイル

```
group lab1 {
    host falcon { jobs = 10 , path = "/set/dist/sparc-S2/bin" }
    host hawk { path = "/opt/SUNWspro/bin" }
}
```

`.dmakerc` ファイル内のグループ名とホスト名は、二重引用符で囲むことができます。二重引用符で囲むことで、グループ名に使用できる文字の種類が増え、英字だけでなく、数字も使用できるようになります。たとえば、次に示すように、グループ名が数字で始まる場合は、二重引用符で囲んでください。

コード例 C-4 特殊文字を使用した `.dmakerc`

```
group "123_lab" {
    host "456_hawk" { path = "/opt/SUNWspro/bin" }
}
```

構築サーバー

分散構築に参加する各構築サーバーには、`/etc/opt/SPROdmake/dmake.conf` というファイルが存在している必要があります。このファイルは、構築サーバー構成ファイルで、特定の構築サーバーに全 `dmake` ユーザーが分散できる `dmake` ジョブの最大数を指定します。また、すべての `dmake` ジョブを実行する際の優先順位 `nice` を指定できます。

注 - 構築サーバーに `/etc/opt/SPROdmake/dmake.conf` ファイルがない場合は、そのサーバー上で `dmake` ジョブを実行することはできません。

コード例 C-5 に `/etc/opt/SPROdmake/dmake.conf` ファイルの例を示します。この例では、(`dmake` ユーザーが全体として) 1 つの構築サーバー上で実行できる `dmake` ジョブの最大数を 8 個に設定しています。

コード例 C-5 `dmake.conf` ファイル

```
max_jobs: 8
nice_prio: 5
```

構築サーバーとして使用するマシンは、次の条件を満たしている必要があります。

- `dmake` ホスト (使用するマシン) から、パスワードの入力を求められずに `rsh` を使用して、構築サーバー上でコマンドを遠隔実行できること。
`rsh(1)` のマニュアルページを参照してください。以下に例を示します。

```
% rsh build-server which dmake
/opt/SUNWspro/bin/dmake
```

- その構築サーバーから、`dmake` ソフトウェアがインストールされている `bin` ディレクトリにアクセスできること。
すべての構築サーバーが同じ `dmake` インストールディレクトリを共有するようにしてください。`share(1M)` および `mount(1M)` のマニュアルページを参照してください。

- `dmake` から見て、デフォルトで、`dmake` 実行可能ファイルへの論理パスが構築サーバー上と `dmake` ホスト上とで同じであること。
このデフォルト設定は、実行時構成ファイルのホストエントリの属性としてパス名を指定することによって変更できます。以下に例を示します。

```
group sparc-cluster {
  host wren    { jobs = 10 , path = "/export/SUNWspro/bin" }
  host stimpay { path = "/opt/SUNWspro/bin" }
}
```

- 構築するソースの階層構造が、構築サーバーで同じ名前でもマウントされ、アクセスできること。

dmake ユーティリティとメイクファイル

分散メイクを実行するには、標準的な `make` ユーティリティの代わりに `dmake` という実行可能ファイルを使用します。したがって、`dmake` を使用する前に、Solaris の `make` ユーティリティについて理解しておく必要があります。`make` ユーティリティの詳細については、『プログラミングユーティリティ』(<http://docs.sun.com> Web サイトで参照可能) および `make(1)` のマニュアルページを参照してください。これまでに `make` ユーティリティを使用したことがある場合は、ほとんど変更作業を伴わずに、`dmake` に移行できます。

この節では、`dmake` でのさまざまな問題の解決に役立つ方法と例を紹介します。必ずしもこれらの方法が最善の解決策であるとは限りませんが、できるだけわかりやすく機能が説明されています。

手順が複雑になるに従って、これを実現するメイクファイルも複雑になります。この節では一般的なコード開発の作業を進める上での問題点を取り上げ、`dmake` による簡単な解決方法について、例を紹介しながら説明します。

プロジェクトの最初の段階からメイクファイルテンプレートを利用するようにしてください。メイクファイルテンプレートから作成することによって、実際のメイクファイルが理解しやすく、統合、保守、再利用が容易になります。

ターゲットの並列構築

大規模なソフトウェア開発プロジェクトは、それぞれ独立した多数のモジュールで構成されるのが一般的です。これらのモジュールを、それぞれ分散させ、並列的に構築できます。こうした並列処理により、大規模な開発プロジェクトを構築するのに必要な時間が大幅に省けます。

`dmake` は、構築するターゲットが与えられると、そのターゲットに関連する依存関係を調べ、同期の取れていない依存関係も構築します。この依存関係の構築では、依存関係の依存関係の構築を伴うこともあります。ジョブを分散する際、`dmake` は起動可能なすべてのターゲットを起動します。それらのターゲットが終了すると、`dmake` はその他のターゲットを起動します。デフォルトでは、入れ子構造になった `dmake` の呼び出しは同時には実行されません。ただし、この設定は変更可能です (詳細については、99 ページの「並列処理に対する制限」を参照)。

`dmake` では複数のターゲットが並列に構築されるため、各構築プロセスの出力は同時に生成されます。各コマンドからの出力が混ざらないよう、`dmake` では各構築の出力を個別に収集します。また `dmake` は、コマンドを実行する前に、そのコマンドの内容を表示します。実行したコマンドによって出力、警告、またはエラーが生成されると、`dmake` ユーティリティは該当するコマンドの出力全体を表示します。後から開始したコマンドが先行するコマンドよりも先に終了することもあるので、出力の順序が予想と異なる場合があります。

メイクファイルに対する制限事項

複数のターゲットを同時に構築する場合は、メイクファイルに多少の制限が伴います。たとえば、暗黙の依存関係の順序に依存しているメイクファイルは、並列構築を行なったとき正しく機能しないことがあります。また同じファイルの修正を行うターゲットがあり、これらファイルが2つの異なるターゲットで同時に変更されると、メイクファイル中のターゲットで問題が起きることがあります。この節では、こうした問題の例を取り上げます。

依存関係リスト

ターゲットの並列構築をするときは、依存関係リストが正確であることが重要なポイントになります。たとえば、2つの実行可能ファイルが同じオブジェクトファイルを共有しているときに、一方のファイルでしか依存関係が指定されていないと、並列構築がエラーになる可能性があります。次のメイクファイルの例を考えてみます。

```
all: prog1 prog2
prog1: prog1.o aux.o
      $(LINK.c) prog1.o aux.o -o prog1
prog2: prog2.o
      $(LINK.c) prog2.o aux.o -o prog2
```

逐次構築では、ターゲット `aux.o` は `prog1` に依存するファイルとして構築され、`prog2` の構築時に利用されます。これに対し、並列構築では、`prog2` のリンクが、`aux.o` の構築前に開始されることがあり、その場合は、不正になります。make の `.KEEP_STATE` 機能で検出できる依存関係も一部ありますが、上記のような関係はその対象にはなりません。

依存関係リストにおける依存関係の順序指定

次に、暗黙の順序にもとづく依存関係がある場合は、問題がさらに複雑になるという例を示します。たとえば、あるシステムのヘッダーをすべて作成してから構築を開始する場合は、すべての要素がこの構築に依存していなければなりません。このため、メイクファイルがさらに複雑になり、新しいターゲットをメイクファイルに追加した場合などにエラーが起きる可能性が高くなります。この場合は、メイクファイルの中で特殊な `.WAIT` ターゲットを指定して、暗黙の依存関係があることを明示できます。dmake により依存関係リストの中に `.WAIT` ターゲットが検出されると、これまでの依存関係の処理を一時停止してから、以降の依存関係の処理を実行するようになります。1つの依存関係リストに、複数の `.WAIT` ターゲットを入れることもできます。次に、`.WAIT` を使用して、まずヘッダーを作成してから、以降の処理を続行する例を示します。

```
all: hdrs .WAIT libs functions
```

`.WAIT` ターゲットに適用する空の規則をメイクファイルに追加することによって、make ユーティリティでも、このメイクファイルを使用できます。

ファイルの同時修正

ターゲットの構築プロセスでは、同時に同じファイルを修正しないように注意する必要があります。この問題は、さまざまな形で起きる可能性があります。一時ファイルを使用する新しい接尾辞の規則を定義する場合、一時ファイルの名前は、ターゲットごとに異ならなければなりません。この場合は、`$@` や `$*` の動的マクロを利用できます。たとえば、ソースファイルをコンパイルする前に `.c` ファイルに修正を加える場合、`.c.o` の規則は次のように定義できます。

```
.c.o:  
    awk -f modify.awk $*.c > $*.mod.c  
    $(COMPILE.c) $*.mod.c -o $*.o  
    $(RM) $*.mod.c
```

ライブラリの同時更新

並列処理でもう1つの問題になる可能性があるのは、ライブラリを作成するときのデフォルトの規則です。すなわち、ライブラリという固定ファイルが変更される場合があるということです。本来適用されるべきでない `.c.a` 規則によって、`dmake` がそれぞれのオブジェクトファイルを構築し、このオブジェクトファイルをアーカイブする可能性があります。`dmake` が2つのオブジェクトファイルを同時にアーカイブすると、並列的に更新が行われてアーカイブファイルが壊れることがあります。

```
.c.a:  
    $(COMPILE.c) -o $% $<  
    $(AR) $(ARFLAGS) $@ $%  
    $(RM) $
```

この場合は、各オブジェクトファイルを構築し、その構築が完了した後、すべてのオブジェクトファイルをアーカイブするようにします。正しい接尾辞の規則と対応するライブラリの規則は次のとおりです。

```
.c.a:  
    $(COMPILE.c) -o $% $<  
  
lib.a: lib.a($(OBJECTS))  
    $(AR) $(ARFLAGS) $(OBJECTS)  
    $(RM) $(OBJECTS)
```

複数のターゲット

ファイルの同時更新の問題は、同じ規則を複数のターゲットに対して定義した場合にも起きます。たとえば、lex(1) で使用できるプログラムとヘッダーの両方を構築する yacc(1) というプログラムがあります。1つの規則にもとづいて複数のターゲットファイルを構築する場合は、+記号を使用して各ファイルを1つのグループとして指定する必要があります。特に、並列構築の場合に、このことが重要になります。

```
y.tab.c y.tab.h: parser.y
    $(YACC.y) parser.y
```

この規則は、次のように2つの規則が指定された場合と同等になります。

```
y.tab.c: parser.y
    $(YACC.y) parser.y
y.tab.h: parser.y
    $(YACC.y) parser.y
```

逐次処理の make は、最初の規則に基づいて y.tab.c を作成してから、y.tab.h が最新であり、構築の必要がないと判断します。これに対し、並列構築では、yacc が y.tab.c の構築を完了する前に y.tab.h を調べるため、y.tab.h の構築が必要であると判断し、最初の yacc と並行して別の yacc を起動します。両方の yacc が同じファイル (y.tab.c と y.tab.h) に書き込みを行うため、これらのファイルが壊れ、不正になる可能性があります。正しい規則では、次の例に示すように、+ 構文を使用して、両方のターゲットを同じ規則に従って同時に作成するように指示します。

```
y.tab.c + y.tab.h: parser.y
    $(YACC.y) parser.y
```

並列処理に対する制限

メークファイルでファイルの衝突が避けられない場合があります。その1つの例が、Cプログラムから文字列を抽出して共有文字列を生成する `xstr(1)` コマンドです。`xstr` コマンドでは、変更したCプログラムを `x.c` という固定したファイルに書き出し、抽出した文字列は `xs.c` という固定ファイルに書き出します。`xstr` はCファイル単位で実行する必要があるため、次の新しい `.c.o` 規則がよく定義されます。

```
.c.o:
 $(CC) $(CPPFLAGS) -E $*.c | xstr -c -
 $(CC) $(CFLAGS) $(TARGET_ARCH) -c x.c
 mv x.o $*.o
```

この場合、各ターゲットを作成するごとに、`x.c` と `xs.c` という同じファイルに書き出しを行うので、`dmake` ユーティリティは上の規則に基づいてターゲットを同時に構築することはできません。また、使用するファイルを変更することもできません。この場合は、`.NO_PARALLEL:` という特殊なターゲットを使用して、`dmake` に各ターゲットを同時に構築しないように指示できます。たとえば、`.c.o` 規則を使用して構築するオブジェクトを `OBJECTS` マクロで定義している場合は、次のようにして、各ターゲットを逐次構築するよう `dmake` に指示します。

```
.NO_PARALLEL: $(OBJECTS)
```

大部分のオブジェクトを逐次構築する必要がある場合は、依存関係のない `.NO_PARALLEL:` ターゲットを含めることによって、全オブジェクトをデフォルトで強制的に逐次処理するようにした方が、安全で簡単です。並列構築できるターゲットは、次のように `.PARALLEL:` ターゲットに依存するファイルとして記述できます。

```
.NO_PARALLEL:
 .PARALLEL: $(LIB_OBJECT)
```

`dmake` が、さらに別の `dmake` コマンドを呼び出すターゲットを検出すると、並列ではなく、逐次にターゲットが作成されます。これにより、2つの異なる `dmake` によって、同じディレクトリ中の同じターゲットが構築されなくなります。このような問題は、2つの異なるプログラムを並列構築して、両方が同じライブラリにアクセスする場合などに起きます。それぞれの `dmake` の呼び出しで、必ず最新のライブラリを

適用するには、`dmake` を再帰的に呼び出して、ライブラリを構築する必要があります。`dmake` ユーティリティは、コマンド行から `$(MAKE)` マクロが使用されている場合にのみ、入れ子になった呼び出しを認識します。

コマンドを入れ子にしても、衝突することがない確信がある場合は、`.PARALLEL:` 構文を使用して並列処理できます。

メイクファイルの中に、並列的に同時に動作する入れ子のコマンドが多数含まれている場合は、負荷均衡アルゴリズムによって、多数の構築ジョブがローカルのマシンに割り当てられることがあります。この結果、スワップ空間の不足といった別の問題が生じたり、負荷が大きくなったりすることも考えられます。こうした問題が起きた場合は、入れ子にしたコマンドをすべて逐次実行されるようにしてください。

付録 D

sbquery、sb_init、sbtags による ソースのブラウズ

この付録では、次の事項を説明します。

- sbquery ユーティリティ：ソースコードをブラウズするためのコマンド行ユーティリティです。
- ソースファイルに関するデータベース情報が複数のディレクトリに分散している場合の作業方法。
- sbtags コマンド。ソースファイルからブラウズ情報を短時間かつ簡単に収集する手段です。

この付録は、主として、Sun WorkShop 統合プログラミング環境内からも行える作業をコマンド行から行う方法を説明しています。ソースのブラウズに関する概念的な情報については、このマニュアルの第 5 章と オンラインヘルプを参照してください (オンラインヘルプには、どの Sun WorkShop ウィンドウからも、「ヘルプ」メニューからアクセスできます)。

sbquery ユーティリティ

sbquery は、コマンド行から利用できるユーティリティです。デフォルトでは、sbquery は現在の作業用ディレクトリ内のデータベースからシンボルを検索します。別のディレクトリに格納されているデータベースをブラウズする場合は、106 ページの「sb_init ファイルと sb_init コマンド」を参照してください。

コマンド行から問い合わせを発行するには、`sbquery` に続けて、コマンド行のオプションと引数、検索対象のシンボルを入力します。

```
% sbquery [options] symbols
```

`sbquery` は、`symbols` に一致するものに関する情報、すなわち、シンボルを含むファイル、行番号、シンボルを含む関数、シンボルを含むソース行を一覧表示します。

sbquery のオプション

`sbquery` のコマンド行オプションを一覧表示するには、シェルプロンプトで `sbquery` と入力します。表 D-1 に、このコマンドのオプションを示します。`sbquery(1)` のマニュアルページも参照してください。

表 D-1 sbquery のオプション

引数	説明
<code>-pattern symbol</code>	先頭ダッシュ (-) など、特殊な文字を含むシンボルを問い合わせます。このオプションでは、コマンド行オプションのようなシンボルに対する問い合わせが可能です。たとえば、シンボル <code>-help</code> を問い合わせる場合は、通常の <code>-help</code> オプションと区別するために使用します。
<code>-break_lock</code>	データベースのロックを解除します。索引ファイルの更新が異常終了した後に問い合わせを発行すると、データベースがロックされているというメッセージが表示され、この引数が必要になることがあります。このオプションを使用した後は、データベースの状態が不整合になる可能性があります (整合性を確保するには、データベースディレクトリを削除してから、プログラムを再コンパイルします)。
<code>-files_only</code>	検索対象のシンボルが含まれるファイルだけを表示します。
<code>-help</code>	<code>sbquery</code> コマンドの形式を表示します。 <code>sbquery</code> をオプションやシンボルなしで入力しても同じです。
<code>-help_focus</code>	ディレクトリ内の特定の種類のプログラムのみを問い合わせるためのフォーカスオプションのリストを表示します (表 D-3 を参照)。このオプションを使用して、プログラムや関数といった特定のコード単位に限定した問い合わせを発行できます。

表 D-1 sbquery のオプション (続き)

引数	説明
<code>-help_filter language</code>	<code>-help_filter</code> だけを指定した場合は、フィルタオプションで利用できる言語の一覧を表示します。 <code>-help_filter language</code> の形式で入力した場合は、指定された言語で利用できるフィルタオプションの一覧を表示します。プログラム内の使用法に基づいてシンボルを探すには、フィルタオプションを使用します (表 D-2 を参照)。
<code>-max_memory size</code>	索引ファイルの構築時、sbquery が一時ファイルを使用する前に割り当てる必要のあるメモリー容量の概算値 (メガバイト単位) を設定します、
<code>-no_case</code>	問い合わせで大文字と小文字を区別します。
<code>-no_source</code>	一致したもののファイル名と行番号だけ表示し、一致したものを含むソース行を表示しません。
<code>-no_update</code>	コンパイルの後に問い合わせを発行するときに索引ファイルを再構築しません。このオプションを指定しないでコンパイルまたは再コンパイルの後に問い合わせを発行すると、データベースによって問い合わせの更新と処理が行われます。
<code>-o file</code>	問い合わせの出力を <i>file</i> に送信します。
<code>-show_db_dirs</code>	問い合わせを発行したときに読み取られたデータベースディレクトリのリストを表示します。リストに含まれるのは、現在の作業ディレクトリ内のデータベースディレクトリと、sb_init ファイルの import または export コマンドで指定されたその他すべてのデータベースディレクトリです。
<code>-symbols_only</code>	検索パターン内のパターンに一致するシンボルのリストを表示します。問い合わせでワイルドカードを使用する場合に役立ちます。
<code>-version</code>	現在のバージョン番号を表示します。

表 D-1 sbquery のオプション (続き)

引数	説明
-sh_pattern	ワイルドカードを含む問い合わせを発行するときにシェル形式の表現を使用します (デフォルト)。同じコマンド行で別のパターン検索を実行する場合に、このオプションを指定します (シェル形式のパターン検索については、sh(1) を参照)。
-reg_expr	ワイルドカードを含む問い合わせを発行するときに正規表現を使用します。このオプションを指定しないと、シェル形式のパターンとみなされます。
-literal	問い合わせにワイルドカード表現を使用しません。別のワイルドカード方式からメタキャラクタを含む文字列を検索する場合に役立ちます。

検索を絞り込むためのオプションには、フィルタオプション (表 D-2) とフォーカスオプション (表 D-3) の 2 種類があります。

表 D-2 のフィルタオプションは、プログラム内でシンボルがどのように使用されているかに応じて、シンボルを検索するときに使用します。たとえば、フィルタオプションを使用することができます、検索対象を変数の宣言に限定できます。

```
% sbquery -help_filter language
```

表 D-2 言語フィルタオプション

フィルタオプション	説明
ansi_c	C
sun_as	アセンブリ言語
sun_c_plus_plus	C++
sun_f77	FORTRAN 77

表 D-3 のフォーカスオプションは、検索範囲を、特定のプログラム、関数、ライブラリなどの、特定のクラスのコードに限定します。

```
% sbquery focus-option symbol
```


表 D-3 フォーカスオプション

フォーカスオプション	説明
<code>-in_program program</code>	検索範囲を <i>program</i> (プログラム) に限定
<code>-in_directory directory</code>	検索範囲を <i>directory</i> (ディレクトリ) に限定
<code>-in_source_file source-file</code>	検索範囲を <i>source-file</i> (ソースファイル) に限定
<code>-in_function function</code>	検索範囲を <i>function</i> (関数) に限定
<code>-in_class class</code>	検索範囲を <i>class</i> (クラス) に限定
<code>-in_template template</code>	検索範囲を <i>template</i> (テンプレート) に限定
<code>-in_language language</code>	検索範囲を <i>language</i> (言語) に限定

注 - フォーカスオプションを 2 つ以上指定して、一致するものが複数見つかった場合は、一致したものが 1 つだけ返されます。

環境変数

環境変数は、`sbquery` (および Sun WorkShop 統合プログラミング環境のソースブラウザ) の動作に影響する情報を提供します。

表 D-4 環境変数

変数	説明
HOME	ログインディレクトリの名前
PWD	現在のディレクトリのフルパス名
SUNPRO_SB_ATTEMPTS_MAX	ロックされたデータベースに対して索引ビルダーがアクセスを試みる最大回数
SUNPRO_SB_EX_FILE_NAME	<code>sun_source_browser.ex</code> ファイルの絶対パス名
SUNPRO_SB_INIT_FILE_NAME	<code>sb_init</code> ファイルの絶対パス名。 <code>sb_init</code> については、106 ページの「 <code>sb_init</code> ファイルと <code>sb_init</code> コマンド」を参照してください。

sb_init ファイルと sb_init コマンド

この節では、ソースファイルに関するデータベース情報が複数のディレクトリに格納されている場合の作業方法について説明します。デフォルトでは、現在の作業用ディレクトリにデータベースが作成され、問い合わせを発行するとそのデータベースが検索されます。

Sun WorkShop のソースブラウザ機能、コンパイラ、sbtags は、sb_init テキストファイルを使用して、ソースブラウザ用データベースの構造に関する制御情報を取得します。ソースファイルに関するデータベース情報が複数のディレクトリに分散している場合は、sb_init を使用してください。

sb_init ファイルは、SunWS_config ディレクトリに置くようにしてください。このディレクトリから、ソースブラウザ機能やコンパイラ、sbtags も実行するようにします。これらのツールは、sb_init ファイルを探すとき、現在の作業用ディレクトリを検査します。

デフォルトでは、現在の作業ディレクトリに対して sb_init ファイルの検索が行われます。別のディレクトリで sb_init ファイルを検索するように Sun WorkShop 統合プログラミング環境およびコンパイラに指示するには、環境変数 SUNPRO_SB_INIT_FILE_NAME を /<検索するディレクトリのパス>/sb_init に設定します。

sb_init コマンドを使用するには、sb_init ファイルにコマンドを追加します。sb_init ファイルでは、次のコマンドのみ使用できます。

表 D-5 sb_init で使用できるコマンド

コマンド	説明
import	現在の作業用ディレクトリ以外のディレクトリにデータベースを読み込みます。
export	特定のソースファイルと関連付けられたデータベース構成要素ファイルを、コンパイラの現在の作業ディレクトリ以外のディレクトリに書き込みます。このコマンドには、import コマンドの機能もあります。

表 D-5 sb_init で使用できるコマンド (続き)

コマンド	説明
replacepath	データベースで検索するファイル名のパスの変更方法を指定します。データベースを移動する。
automount-prefix	プログラムのコンパイルに使用されなかったマシン上でソースファイルをブラウズできます。
cleanup-delay	索引の構築から関連するデータベースのガベージコレクタまでの許容時間を制限します。

import

```
% import pathname
```

このコマンドは、Sun WorkShop のソースブラウズモードで、現在の作業用ディレクトリ以外のディレクトリにデータベースを読み込むことができます。このため、異なるディレクトリにデータベースを分散できます。

たとえば、プロジェクト A のプログラマがプロジェクト B のディレクトリに書き込むこともその逆もできないように、管理権限を設定しようとしています。この場合、プロジェクト A とプロジェクト B は、それぞれ独自のデータベースを保持する必要があります。どちらのデータベースも、他方のプロジェクトのプログラマからは読み取りしかできず、書き込みはできません。

export

```
% export prefix into path
```

このコマンドは、コンパイラと shtags が、Sun WorkShop のソースブラウズモードとコンパイラで使用される現在の作業用ディレクトリ以外のディレクトリに、ソースファイルのデータベースコンポーネントファイルを書き込むことを可能にします。

prefix で始まる完全パス名の付いたソースファイルをコンパイラが処理した場合、生成されるブラウザ用データベース (.bd) ファイルは必ず *path* に格納されます。

export コマンドには *path* の import コマンドが暗黙的に含まれているので、エクスポートされたデータベースコンポーネントは Sun WorkShop のソースブラウザモードで自動的に読み込まれます。

export コマンドは、`/usr/include` の `#include` など、同じファイルと関連付けられたデータベースファイルを単一のデータベースに配置してディスク空間を節約しながら、各プロジェクト用のデータベースを個々に保持することを可能にします。

`sb_init` ファイルに複数の export コマンドを含める場合は、範囲の広い順にコマンドを並べる必要があります。コンパイラは、`sb_init` ファイルに記述されている順序で export コマンドを読み取ります。

replacepath

```
% replacepath from-prefix to-prefix
```

このコマンドは、ソースブラウザ用データベース内のパス名を変更する方法を指定します。

一般に、*from-prefix* はオートマウンタのマウントポイント (オートマウンタが実際にファイルシステムをマウントする場所のパス名) であり、*to-prefix* はオートマウンタのトリガーポイント (開発者が使用する既知のパス名) です。

オートマウンタの使用方法は柔軟性に富んでおり、ホストによって異なることがあります。

オートマウンタの設定を元に戻すには、デフォルトの `replacepath` コマンドを使用します。

```
% replacepath /tmp_mnt
```

このためには、コマンドの第 1 引数にマウントポイント、第 2 引数にトリガーポイントを指定する必要があります。

automount-prefix

```
% automount-prefix mount-point trigger-point
```

automount-prefix コマンドを使用して、プログラムのコンパイルに使用されなかったマシン上でソースをブラウズできます。このコマンドは、automount-prefix のパス変換がコンパイル時に行われてデータベースに書き込まれる点を除けば replacepath コマンドと同じです。

automount-prefix コマンドは、データベースに格納されている名前からのオートマウンタの接頭辞を削除するかを定義します。オートマウンタがファイルシステムをマウントするディレクトリは *mount_point* です。*trigger_point* は、エクスポートされたファイルシステムにアクセスするための接頭辞であり、デフォルトは / です。

データベース内のパスで問題が発生した場合、automount_prefix コマンドと replacepath コマンドのパス変換情報を使用して、ブラウズ中にソースファイルが検索されます。

一見すると、両方のパスの検索は不可能と思われます。コンパイラの実行中に作成されたブラウザ用データベースには、各ソースファイルの絶対パスが格納されています。絶対パスがコンピュータ間で統一されていないと、問い合わせに回答するときに Sun WorkShop がソースファイルを表示できません。

この問題は、次のいずれかの方法で回避できます。

- すべてのコンピュータですべてのソースファイルが同じマウントポイントにマウントされるようにする
- オートマウントされたパスでプログラムをコンパイルする。そうしたパスに対する参照があると、オートマウンタは別のコンピュータのファイルシステムを自動的にマウントします。

オートマウンタの設定を元に戻すには、デフォルトの automount-prefix コマンドを使用します。

```
% automount-prefix /tmp-mnt /
```

automount-prefix コマンドが指定されていない場合に限り、デフォルトの規則が生成されます。

オートマウンタの使用方法に関する詳細は、`automount(1M)` のマニュアルページを参照してください。

`cleanup-delay`

このコマンドは、索引の再構築から関連するデータベースのガベージコレクトまでの時間を制限します。制限時間が経過すると、コンパイラは自動的に `sbcleanup` を呼び出します。デフォルトでは 12 時間です。

`sbtags` ユーティリティ

`sbtags` コマンドを使用して、ソースファイルからブラウザ情報を簡単に収集できます。この方法を利用すると、完全にコンパイルされていないプログラムに対しても最低限のブラウザ情報を収集できます。詳細は、`sbtags(1)` のマニュアルページを参照してください。

`sbtags` コマンドは、コンパイルで入手可能な情報の一部を収集します。ただし、簡略化された情報では、一部のブラウザ機能に制限があります。`sbtags` で生成されたデータベースでは、関数や変数に問い合わせを実行したり、関数の呼び出しをグラフィカルに表示したりできます。

現在、`sbtags` で生成されるタグデータベースには次の制限があります。

- ローカル変数に関する問い合わせを発行できない
- クラスをブラウザできない
- クラスの関係をグラフィカル表示できない
- 複雑な問い合わせを発行する機能に制限がある
- 問い合わせを限定する機能に制限がある
- コンパイルされた情報よりも信頼性が低い

ファイルが変更されても、しばしば、再度読み取って変更をデータベースに取り込む必要がないことがあります。

`sbtags` データベースは、ソースファイルの字句解析に基づいています。このデータベースは、すべての言語構造を正しく識別するとは限りませんが、コンパイルされないファイルでも動作し、再コンパイルよりも高速に実行されます。

sbtags は、型と関数の定義を認識し、関数呼び出しの情報を収集します。関数呼び出しの情報以外は収集されません。特に、複雑な問い合わせに関する他の意味情報は収集されません。

sbtags の機能は、呼び出しのグラフィカル表示機能を除けば、ctags や etags と類似しており、定義とグラフィカル表示に関するデータベースへの直接の問い合わせと、パターン検索による問い合わせを組み合わせたことができます。

sbtags で生成されたデータベースには、次の特徴があります。

- クラスブラウザとクラスグラフィの機能を使用できない。
- データベースには、シンボルと文字列に関する情報がすべて格納されているとは限らない。関数、クラス、型、関数呼び出しに関する情報は格納されている。
- sbtags プログラムはコンパイラよりも高速に動作するため、時間の節約になる。
- データベースのサイズは、ソースコードのサイズよりもずっと小さい。
- データベースの内容については、意味上の正確さが保証されない (sbtags プログラムが実行するのは単純な構文解析と意味解析だけであり、エラーが発生することもあるため)。
- コードが不完全であるか意味に誤りがあるためにソースコードをコンパイルできなくても、データベースが生成される。

sbtags を使用してブラウザデータベースを生成するには、コマンド行から次のコマンドを入力します。

```
% sbtags file
```

file は、データベースを作成するファイルです。詳細は、sbtags(1) のマニュアルページを参照してください。

用語集

高速実行モード

デバッグモードの1つで、デバッグをバックグラウンドで動作させながら、プログラムを実行することが可能なモード。異常終了した場合、プログラムは保存される。「デバッグモード」も参照。

構築コマンド

make ユーティリティとして起動するコマンド。起動された make ユーティリティはメイクファイルを読み取り、構築ターゲットを構築する。

構築 (make) のターゲット

make ユーティリティがメイクファイルに含まれる指示 (規則) から構築方法に関する情報を得るオブジェクト。

構築ディレクトリ

構築プロセスが起動されたディレクトリ。このディレクトリはメイクファイルのデフォルトのディレクトリでもある。

実行時パラメタ

プログラムのデバッグ中にプログラムに渡されるプログラムの引数、プログラムを実行するディレクトリ、環境変数のこと。

ソースブラウザモード

「ブラウザ」ウィンドウに用意されているモードの1つで、ソースブラウザオプション (-xsb) を使ってソースファイルをコンパイルしたときに作成されるデータベースに基づいて、プログラムに定義されている特定のシンボルに一致するもののすべてを見つけることができるモード。「パターン検索モード」も参照。

ターゲット

構築可能なオブジェクトのこと。

データ表示

プログラム実行中の式の値の変化を観察できる、デバッグサービスの機能。

データ履歴

プログラムのデバッグ中に式を評価したり、変数の値を変更したりできる、デバッグサービスの機能。

デバッグセッション

1つのプログラムに関連付けられたデバッグプロセス。セッションマネージャーを使用して、同時に多数のプログラムをデバッグできる。

デバッグモード

デバッグモードの1つで、デバッグサービスの全機能を使用してプログラムをデバッグすることが可能なモード。「高速実行モード」を参照。

パターン検索モード

「ブラウザ」ウィンドウに用意されているモードの1つで、ソースコードから、コメント内のテキストなどの任意の文字列を検索できるモード。「ソースブラウザモード」も参照。

ピックリスト

「メニューピックリスト」を参照。

プロジェクト

実行可能ファイル、静的ライブラリ / アーカイブ、共有ライブラリ、FORTRAN アプリケーション、複合アプリケーション、ユーザーメークファイルアプリケーションの構築に使われるファイルやコンパイラ、デバッグ、その他構築関連オプションのリスト。

分散メイク (dmake)

make ユーティリティの機能拡張版で、構築作業を複数の作業に分割して、複数の CPU やワークステーションに分散するユーティリティ。

メイクファイル

どのような構造でソースおよびオブジェクトファイルが他のファイルに依存しているかを make ユーティリティに示すエントリから構成されるファイル。このファイルには、それらファイルのコンパイルとリンクに必要なコマンドも含まれる。

メニューピックリスト

Sun WorkShop のメニューに表示される、最近使ったファイルやターゲット、プログラム、プロジェクト、実験のリスト。このリストから、最近使った項目に簡単にアクセスできる。

索引

D

- dbx コマンド, 25
 - 「dbx コマンド」 ウィンドウ, 25, 33
- dmake
 - 基本概念, 89
 - コマンド, 99
 - さらに呼び出す, 99
 - ジョブ、制御, 89
 - ホスト, 89
 - メイクファイル, 94
- dmake.conf ファイル, 20, 93
- .dmakerc ファイル, 90

G

- g0 オプション, 26
- g オプション, 26

H

- HOME
 - 環境変数, 105

M

- make ユーティリティ, 15, 81

N

- .NO_PARALLEL
 - ターゲット, 99

P

- PATH 環境変数、設定, xix
- PWD
 - 環境変数, 105

S

- sb_init ファイル, 106
- sb_init ファイルコマンド
 - automount-prefix, 109
 - cleanup-delay, 110
 - export, 107
 - import, 107
 - replacepath, 108
- sbquery
 - オプション, 102
 - 環境変数, 105
 - 言語フィルタオプション, 104
 - 最大メモリ設定, 103
 - 索引ファイルを再構築しない, 103
 - シェル形式の表現を使用, 104
 - 出力をファイルに送信, 103
 - シンボルのみ表示, 103

正規表現を使用, 104
ソースブラウザ, 101
バージョン番号を表示, 103
フォーカスオプション, 105

sbtags コマンド, 110

Sun WorkShop TeamWare, 6

Sun WorkShop ツール、アクセス, 13

Sun WorkShop の起動, 7

「Sun WorkShop へようこそ」ダイアログボックス, 8

SUNPRO_SB_ATTEMPTS_MAX

環境変数, 105

SUNPRO_SB_EX_FILE_NAME

環境変数, 105

SUNPRO_SB_INIT_FILE_NAME

環境変数, 106, 105

V

Visual GUI 構築ツール, 6

W

.WAIT ターゲット, 96

い

依存関係, 96

イベント、定義, 33

う

ウィンドウ

「dbx コマンド」, 25

「クラスグラフ」, 47

「クラスブラウザ」, 49

「構築」, 17

「コールグラフ」, 45

「データ表示」, 31

「デバッグ」, 25

「ブレークポイント」, 30, 33

「呼び出し元 - 呼び出し先」, 56

メイン, 11

え

エディタ, 1

オプション, 12

デフォルト, 12

お

オートマウントされたパスでプログラムをコンパイルする, 109

オプション

-g, 26

-g0, 26

起動, 12

テキストエディタ, 12

デバッグ, 29

プロジェクト, 13

か

「活動状態セッション」ダイアログボックス, 35

環境変数, 28

HOME, 105

PWD, 105

PWDSUNPRO_SB_ATTEMPTS_MAX, 105

sbquery, 105

SUNPRO_SB_EX_FILE_NAME, 105

SUNPRO_SB_INIT_FILE_NAME, 106, 105

構築, 21

「環境変数」ダイアログボックス, 21

関数

ステップイン, 30

ステップオーバー, 30

ステップ終了, 30

停止位置, 34

関数呼び出し、グラフ表示, 45

き

起動オプション, 12
共有プロジェクトファイルの情報, 9

く

「クラスグラフ」ウィンドウ, 47
クラスのブラウザ, 48
「クラスブラウザ」ウィンドウ, 49
グラフィカルユーザーインタフェース、設計, 6
グラフ表示
 関係のブラウザ, 45
 関数呼び出し, 45
 サブルーチン呼び出し, 45
グリフ、「ファイルマージ」ウィンドウ, 60
クロックに基づくプロファイル, 54, 56

こ

構成ファイル、実行時, 90
高速実行モード、デバッグ, 26
構築
 エラー, 21
 オプション, 20, 84
 環境変数, 21
 サーバー, 89
 指定されたプロジェクト, 18
 終了, 23
 ディレクトリ, 15
 デフォルト値を使用した, 19
 独自の値, 20
 プロジェクトターゲット, 10, 18
構築 make ターゲット、定義, 16
「構築」ウィンドウ, 17
「構築オプション」ダイアログボックス, 20
コード
 ステップ移動, 29
コードのトレース, 33
「コールグラフ」ウィンドウ, 45
コンパイラ, 3

コンパイラ生成ブラウザデータベース, 43
コンパイラ、アクセス, xx

さ

サブルーチン呼び出し、グラフ表示, 45

し

シェルプロンプト, xix
式の値、観察, 32
式のパルーン評価, 31
実験ファイル, 53
実行時検査, 32
実行時パラメータ
 実行ディレクトリ, 28
 プログラム引数, 28
自動マージ, 62
終了
 構築, 23
 デバッグ, 37
 ブラウザ, 51
出力ファイルを保存, 63
書体と記号について, xviii
白抜きのグリフ, 61

す

スタック区画, 34
ステップ
 イン, 30
 オーバー, 30
 関数の終了, 30
 コードの移動, 29
 ソース行を1行先へ進める, 30
 ソース行を1行先へ進める, 30
「スレッド」タブ, 35

せ

制限, 99

正符号 (+), 61

設定

ウインドウの色やフォント, 11

「セッション」タブ, 35

設定

ウインドウの色やフォント, 12

相違オプション, 62

デフォルトのエディタ, 12

ファイルマージオプション, 63

ブレークポイント, 30

そ

相違

間を移動, 61

オプション, 62

解決, 60

自動的に解決, 62

次, 60

定義, 60

テキストファイル間の, 60

残る, 60

前, 60

相違間を移動, 61

相違の解決、定義, 60

相違を自動的に解決, 63

ソースコード管理ツール, 6

ソースファイルのマージ, 57

ソースブラウザ

sbquery, 101

コンパイルされないプログラム, 110

特殊文字, 43

複数ディレクトリ, 44

複数のマシンから, 109

ソースブラウザモード, 42

ソースブラウザ用データベース, 43

コンパイラ生成ブラウザ, 43

タグ生成, 43

パス名を変更, 108

読み込み, 107

ロックの解除, 102

書き込み, 107

祖先ファイル, 57

た

ターゲット

Sun WorkShop, 15

定義, 82

複雑なプロジェクト, 16

複数, 98

並行構築, 95, 96

ユーザーメークファイル, 16

「ターゲットの新規定義」ダイアログボックス, 19

「ターゲットの編集」ダイアログボックス, 19

ダイアログボックス

Sun WorkShop へようこそ, 8

「活動状態セッション」, 35

「環境変数」, 21

「構築オプション」, 20

「ターゲットの新規定義」, 19

「ターゲットの編集」, 19

「テキストエディタのオプション」, 12

「デバッグオプション」, 29

「メークのマクロ」, 21, 86

タグデータベース, 43

制限, 110, 111

定義, 110

縦棒 (|), 61

タブ

「スレッド」, 35

「セッション」, 35

「データ表示」, 31, 32

「データ履歴」, 31

つ

ツール, 13

次の相違、定義, 60

て

- 停止位置関数, 34
- 「データ表示」ウィンドウ, 31
- 「データ表示」タブ, 31, 32
- データベースのブラウズ, 43
 - 書き込み, 107
 - パス名を変更, 108
 - 読み込み, 107
- データベースブラウズ
 - ロックの解除, 102
- 「データ履歴」タブ, 31
- テキスト, 12
- テキストエディタ, 1
 - 「テキストエディタのオプション」ダイアログボックス, 12
- テキストエディタのデフォルト, 12
- デバッグ
 - 2つのプログラムを横に並べる, 36
 - オプション, 29
 - 開始, 26
 - 子プロセス, 36
 - 準備, 26
 - セッション、カスタマイズ, 29
 - デバッグモード, 27
 - デフォルト, 29
 - 複数のプログラム, 35
 - ブレークポイントの設定, 30
 - マルチスレッドプログラム, 35
 - 高速実行モード, 26
- 「デバッグ」ウィンドウ, 25
 - スタック区画, 34
 - 「スレッド」タブ, 35
 - 「セッション」タブ, 35
 - 「データ表示」タブ, 32
 - 「データ履歴」タブ, 31
- 閉じる, 37
- 「デバッグオプション」ダイアログボックス, 29
- デバッグモード, 27
- デフォルトのエディタ、設定, 12

と

- 同期遅延データ, 54, 56
- 特殊文字
 - ソースブラウズ, 43
 - パターン検索, 41
- 閉じる
 - 「デバッグ」ウィンドウ, 37
 - ブラウズ, 51
- トレース
 - コード, 33

の

- 残る相違、定義, 60

は

- パターン検索
 - 特殊文字, 41
 - 複数ディレクトリ, 41
- パターン検索モード, 39
- パフォーマンスアナライザ, 53
- パフォーマンスデータ、収集, 32, 54
- パフォーマンスプロファイルツール, 53

ひ

- 標本コレクタ, 32, 53

ふ

- ファイル
 - .dmakerc, 90
 - dmake.conf, 20, 93
 - sb_init, 106
 - 実行時構成, 20
 - 衝突, 99
 - 祖先, 57
 - ファイルマージ出力, 63
 - ファイルマージに読み込む, 57

- マージ, 57
- ファイルの並行修正, 97
- 「ファイルマージ」ウィンドウグリフ, 60
- ファイルマージオプション, 63
- ファイルマージに読み込む, 57
- 複数のターゲット, 98
- 複数のプログラムをデバッグ, 35
- 負符号 (-), 61
- ブラウザデータベース, 43
- ブラウザ
 - オートマウントされたパス, 109
 - クラス, 48
 - グラフ表示の関係, 45
 - 終了, 51
 - 閉じる, 51
- 「ブラウザ」ウィンドウ
 - ソースブラウザモード, 42
 - パターン検索モード, 40
- ブレークポイント, 30
- ブレークポイント、「アクセス時」, 32
- 「ブレークポイント」ウィンドウ, 30, 33
- プログラム引数, 28
- プロジェクト
 - ウィザード, 8
 - 作成, 9
 - 定義, 7
- プロジェクトオプション, 13
- プロジェクトファイルの情報、共有, 9
- 分散メーク, 20, 89

ほ

- パフォーマンスアナライザ, 53

ま

- 前の相違、定義, 60
- マクロ
 - 動的, 97
 - メークファイル, 85

- マニュアルの索引, xxi
- マニュアルページ、アクセス, xix
- マルチスレッドプログラム、デバッグ, 35

め

- メインウィンドウ, 11
- 「メークのマクロ」ダイアログボックス, 21, 86
- メークファイル
 - C++ の例, 83
 - dmake ユーティリティ, 94
 - Fortran 77 の例, 83
 - 作成, 18
 - 制限事項, 95
 - 定義, 16, 82
 - ファイルの衝突, 99
- メークファイル、規則
 - 定義, 82
- メークファイルに対する制限事項, 95
- メークファイルマクロ, 85
 - 定義, 21
 - 無効, 86

ゆ

- ユーザーメークファイル
 - ターゲット, 16
 - プロジェクト, 19, 20

よ

- 呼び出しスタック
 - 1 レベル上に移動, 34
 - 検査, 34
 - 現フレームまでポップ, 34
 - 停止した関数をスタックから削除, 34
 - 複数のフレームを削除, 34
 - ポップ, 34
 - 1 レベル下に移動, 34
- 「呼び出し元 - 呼び出し先」ウィンドウ, 56

ら

- ライブラリの更新、並行, 97
- ライブラリをアーカイブする, 97

り

- リソース
 - Sun WorkShop ウィンドウ, 65, 46, 47
 - テキストエディタ, 65
 - 変更, 66
 - 編集可能, 67
- リソースファイル
 - ESERVE, 65
 - WORKSHOP, 65

ろ

- ロック lint, 6
- ロックされていないブラウザ用データベース, 102

