SYBASE®

プログラマーズ・ガイド補足

Open Client/Server™

12.5.1

UNIX 版

ドキュメント ID：DC35453-01-1251-01
改訂：2003 年 11 月

マニュアルの注文

マニュアルの注文を承ります。ご希望の方は、サイベース株式会社営業部または代理店までご連絡ください。マニュアルの変更は、弊社の定期的なソフトウェア・リリース時にのみ提供されます。

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目次

はじめに

Sybase® Open Client/Server™ 製品は、アプリケーションと任意のデータ型をともに使用できる、プログラミング・インタフェースのセットです。次の製品が含まれます。

- Open Client DB-Library™/C
- Open Client Client-Library™/C
- Open Server Server-Library/C
- Open Client Embedded SQL™/C
- Open Client Embedded SQL/COBOL

これらの各製品には、製品の詳細を説明する独自のリファレンス・マニュアルがあります。このマニュアルの目的は、製品マニュアルを補足することです。このマニュアルは、すべての Open Client/Server 製品について、プラットフォームに関連した問題を説明します。

このマニュアルでは、次に示す UNIX プラットフォームについて説明します。

- HP Tru64 UNIX
- HP 9000 シリーズ HP-UX
- IBM RISC System/6000 AIX
- Linux
- Silicon Graphics IRIX
- Sun Solaris 2.x (SPARC)

対象読者

このマニュアルでは、上記の Open Client/Server 製品を使用するプログラマの方を対象としています。

このマニュアルの内容

この『Open Client/Server プログラマーズ・ガイド補足 UNIX 版』では、UNIX オペレーティング・システムで稼働する Open Client/Server 製品について説明します。Open Client Client-Library または Open Server などの個々の製品については、それぞれの章で説明します。このマニュアルの各章では、次の項目を説明します。

- 実行プログラムの構築
- オンライン・サンプル・プログラムのロケーションと内容などの情報

付録では、次の項目を説明します。

- Open Client/Server に関連するコマンドとユーティリティの構文、パラメータ、識別子の詳細を説明するリファレンス・ページ
- アプリケーションを構築し、実行するために設定する必要のある環境変数に関する情報

関連マニュアル

各 Open Client/Server 製品にはユーザ・マニュアルのセットがあります。表 1 では、製品とそれに関連するマニュアルを示します。

表 1: 製品マニュアルのリスト

製品	関連マニュアル
Client-Library	『Open Client Client-Library/C リファレンス・マニュアル』 『Open Client/Server Common Libraries リファレンス・マニュアル』 『Open Client Client-Library/C プログラマーズ・ガイド』
DB-Library	『Open Client DB-Library/C リファレンス・マニュアル』
Server-Library	『Open Server Server-Library/C リファレンス・マニュアル』 『Open Client/Server Common Libraries リファレンス・マニュアル』 『Open Client Client-Library/C リファレンス・マニュアル』
ESQL/C	『Embedded SQL/C プログラマーズ・ガイド』
ESQL/COBOL	『Embedded SQL/COBOL プログラマーズ・ガイド』

インストール、ディレクトリ構造、および論理名については、使用しているプラットフォーム用のインストール・ガイドを参照してください。

次の情報については、『Open Client/Server 設定ガイド UNIX 版』を参照してください。

- Open Client アプリケーションとサーバ間で通信するために環境を設定する方法
- Sybase アプリケーションをローカライズする方法

その他の情報ソース

Sybase Getting Started CD、Sybase Technical Library CD、Technical Library Product Manuals Web サイトを利用すると、製品について詳しく知ることができます。

- Getting Started CD には、PDF 形式のリリース・ノートとインストール・ガイドが収録されています。また、その他のマニュアルや、Technical Library CD には含まれない更新情報が収録されることもあります。この CD は製品のソフトウェアに同梱されています。Getting Started CD に収録されているマニュアルを参照または印刷するには、Adobe Acrobat Reader が必要です (CD 内のリンクを使用して Adobe の Web サイトから無料でダウンロードできます)。

- Technical Library CD には製品マニュアルが入っており、この CD は製品のソフトウェアに同梱されています。DynaText リーダー (Technical Library CD に収録) を使用すると、この製品に関する技術情報に簡単にアクセスできます。

  Technical Library のインストールと起動の方法については、マニュアル・パッケージに含まれている『Technical Library Installation Guide』を参照してください。

- Technical Library Product Manuals Web サイトは、Technical Library CD の HTML バージョンで、標準の Web ブラウザを使ってアクセスできます。また、製品マニュアルのほか、EBFs/Updates、Technical Documents、Case Management、Solved Cases、ニュース・グループ、Sybase Developer Network へのリンクもあります。

  Technical Library Product Manuals Web サイトにアクセスするには、Product Manuals (http://www.sybase.com/support/manuals/) にアクセスしてください。

Web 上の Sybase 製品の動作確認情報

Sybase Web サイトの技術的な資料は頻繁に更新されます。

❖ **製品認定の最新情報にアクセスする**

1 Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)

2 左側のナビゲーション・バーから [Products] を選択します。

3 製品リストから製品名を選択し、[Go] をクリックします。

4 [Certification Report] フィルタを選択し、時間枠を指定して [Go] をクリックします。

5 [Certification Report] のタイトルをクリックして、レポートを表示します。

❖ **Sybase Web サイト (サポート・ページを含む) の自分専用のビューを作成する**

MySybase プロファイルを設定します。MySybase は無料サービスです。このサービスを使用すると、Sybase Web ページの表示方法を自分専用にカスタマイズできます。

1 Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)

2 [MySybase] をクリックし、MySybase プロファイルを作成します。

Sybase EBF とソフトウェア・メンテナンス

❖ EBF とソフトウェア・メンテナンスの最新情報にアクセスする

1 Web ブラウザで Sybase Support Page (http://www.sybase.com/support) を指定します。

2 [EBFs/Maintenance] を選択します。すでに Web アカウントをお持ちの場合はユーザ名とパスワードを要求されますので、各情報を入力します。Web アカウントをお持ちでない場合は、新しいアカウントを作成します。サービスは無料です。

3 製品を選択します。

4 時間枠を指定して [Go] をクリックします。

5 EBF/Maintenance レポートを表示するには [Info] アイコンをクリックします。ソフトウェアをダウンロードするには製品の説明をクリックします。

表記の規則

このマニュアルで使用する構文の表記規則は、次のとおりです。

表 2: 構文の表記規則

構文要素	意味
command	太字で表記するものには、コマンド名、コマンドのオプション名、ユーティリティ名、ユーティリティのフラグ、キーワードがある。
variable	変数 (ユーザが入力する値を示す語句) は、斜体で表記する。
{ }	中カッコは、その中から必ず 1 つ以上のオプションを選択しなければならないことを意味する。コマンドには中カッコは入力しない。
[ ]	角カッコは、オプションを選択しても省略してもよいことを意味する。コマンドには角カッコは入力しない。
|	縦線は、中カッコまたは角カッコの中の縦線で区切られたオプションのうち 1 つだけを選択できることを意味する。
,	大カッコまたは角カッコの中のカンマで区切られたオプションをいくつでも選択できることを意味する。複数のオプションを選択する場合には、オプションをカンマで区切る。

次に、上記の構文の表記規則の例を示します。

大カッコ内の縦線は、オプションを 1 つだけ選択することを意味します。

```
{red | yellow | blue}
```

大カッコ内のカンマは、オプションを 1 つ以上選択することを意味します。複数のオプションを選択する場合は、それぞれをカンマで区切ります。

```
{cash, check, credit}
```

角カッコは、オプションのパラメータを示します。角カッコ内の項目を 1 つも選択しなくてもかまいません。

角カッコに項目が 1 つだけある場合は、省略してもかまいません。

```
[anchovies]
```

角カッコ内の縦線は、項目を 1 つ も選択しないか、または 1 つだけ選択することを意味します。

```
[beans | rice | sweet_potatoes]
```

角カッコ内のカンマは、オプションを 1 つも選択しないか、または 1 つ以上のオプションを選択することを意味します。複数のオプションを選択する場合は、それぞれをカンマで区切ります。

```
[extra_cheese, avocados, sour_cream]
```

省略記号 (...) が続いている構文は、最後の単位を何度でも繰り返せることを意味します。次の構文の例では、1 つ以上のプログラム名と拡張子の組を角カッコで囲んでリストできます。

```
cpre -L program[.ext] [program[.ext]]...
```

不明な点があるときは

Sybase ソフトウェアがインストールされているサイトには、Sybase 製品の保守契約を結んでいるサポート・センタとの連絡担当の方 (コンタクト・パーソン) を決めてあります。マニュアルだけでは解決できない問題があった場合には、担当者を通して Sybase のサポート・センタまでご連絡ください。

第 1 章

Open Client Client-Library/C

Open Client Client-Library は、クライアント・アプリケーションの作成に使用するルーチンの集まりです。Client-Library には、サーバにコマンドを送信するルーチンとそれらのコマンドの結果を処理するルーチンが含まれています。さらに、アプリケーション・プロパティの設定、エラー条件の処理、サーバとのアプリケーションの対話に関するさまざまな情報の提供を行うルーチンもあります。

Open Client に含まれている CS-Library は、Open Client アプリケーションや Open Server アプリケーションを作成するために使用できるユーティリティ・ルーチンの集まりです。Client-Library ルーチンは CS-Library で割り付けられる構造体を使用するので、すべての Client-Library アプリケーションには、CS-Library に対する呼び出しが少なくとも 1 つ含まれています。

この章の内容は、次のとおりです。

注意 Open Client 製品に関する補足説明および使用するプラットフォームでの動作については、『リリース・ノート』を参照してください。

一般的な条件

Client-Library のサンプル・プログラムを実行するには、以下の準備が必要です。

- Adaptive Server にアクセスできる必要があります。必要な Adaptive Server のバージョン・レベルについては、それぞれのサンプル・プログラムの説明を参照してください。

- 次の環境変数を設定します。詳細については、「付録 B 環境変数」を参照してください。
  - SYBASE
  - DSQUERY
  - SYBPLATFORM
  - プラットフォーム固有のライブラリ・パス変数
- Adaptive Server に接続できる必要があります。Adaptive Server に接続する方法については、使用しているプラットフォームの『Open Client/Server 設定ガイド』を参照してください。
- サンプル・プログラムを実行する方法の詳細については、*README* ファイルを参照してください。

Client-Library 実行プログラムの構築

この項では、マルチスレッド・アプリケーションを含む Client-Library アプリケーションの構築に必要なライブラリ、およびコンパイルとリンク行について説明します。

表 1-1 は、非スレッド環境ですべての Client-Library 機能を十分に活用するために組み込む必要のあるライブラリを示します。

表 1-1: プラットフォーム固有のライブラリ

プラットフォーム	必要なライブラリ
すべてのプラットフォーム	*libct* − Client-Library (Sybase) *libcs* − CS-Library (Sybase) *libtcl* − トランスポート制御層 (Sybase 内部使用) *libcomn* − 内部共有ユーティリティ・ライブラリ (Sybase 内部使用) *libintl* − 国際化サポート・ライブラリ (Sybase 内部使用)

ネイティブ・スレッドのサポート

Client-Library にはスレッドセーフ・ライブラリが含まれています。開発者はこれを使用して、POSIX スレッドを使用するマルチスレッド・アプリケーションを作成できます。

適切な構文と例については、「マルチスレッド・アプリケーションのコンパイルとリンク行」(7 ページ) を参照してください。

表 1-2 は、マルチスレッド・サポートのためのすべての Client-Library 機能を利用するために組み込む必要があるライブラリを示します。

表 1-2: マルチスレッド・サポート用のプラットフォーム固有ライブラリ

プラットフォーム	必要なライブラリ
すべてのプラットフォーム	*libct_r* − Client-Library (Sybase) *libcs_r* − CS-Library (Sybase) *libintl_r* − 国際化サポート・ライブラリ (Sybase 内部使用) *libm* − UNIX 標準の算術ライブラリ (システム)
Sun Solaris	*libpthread* − スレッド・ライブラリ (システム) *socket* − ソケット・ネットワーク・ライブラリ (システム) *libnsl* − ネットワーク・ライブラリ (システム) *libdl* − 動的ローダ・ライブラリ (システム) *libtcl_r* − トランスポート制御層 (Sybase 内部使用) *libcomn_r* − 内部共有ユーティリティ・ライブラリ (Sybase 内部使用)
HP 9000(8xx)	*libcl* − HP トランスポート制御層 (システム) *libBSD* − BSD ライブラリ (システム) *libc_r* − C 言語リエントラント・ライブラリ *libndbm* − (システム) *libtcl_r* − トランスポート制御層 (Sybase 内部使用) *libcomn_r* − 内部共有ユーティリティ・ライブラリ (Sybase 内部使用)
IBM RS/6000	*libc_r* − C 言語リエントラント・ライブラリ *libpthreads* − スレッド・ライブラリ (システム) *libtcl_r* − トランスポート制御層 (Sybase 内部使用) *libcomn_r* − 内部共有ユーティリティ・ライブラリ (Sybase 内部使用)
Linux	*libtcl_r* − トランスポート制御層 (Sybase 内部使用) *libc_r* − C 言語リエントラント・ライブラリ *libpthreads* − スレッド・ライブラリ (システム) *libtcl_r* − トランスポート制御層 (Sybase 内部使用) *libcomn_r* − 内部共有ユーティリティ・ライブラリ (Sybase 内部使用)

Kerberos サポート

Client-Library バージョン 11.1 以降は、ネットワークを介して通信するときに高度なセキュリティを必要とするアプリケーションに対して Kerberos セキュリティ機能をサポートします。必要な CyberSafe ソフトウェアをインストールして適切な設定作業を実行することによって、Client-Library アプリケーションはこのバージョンでサポートされる次の Kerberos セキュリティ機能を利用できます。

- ネットワーク認証
- 相互認証
- 順序不整合認証
- リプレイの検出
- 機密性

- 整合性

注意 このバージョンでの、HP Tru64 UNIX または SGI プラットフォームに対する Kerberos サポートはありません。このほかのサポートの最新情報については、『リリース・ノート』を参照してください。

Kerberos 機能を利用する Client-Library アプリケーションを作成して実行するには、表 1-3 に示す作業を行う必要があります。

表 1-3: CyberSafe Kerberos サポート用に必要な作業

作業	詳細
次の CyberSafe ソフトウェアをシステムにインストールする。CyberSafe Challenger バージョン 5.2.6 以降、CyberSafe Application Development Kit バージョン 1.0.2	CyberSafe のマニュアルを参照。
CSFC5CCNAME 環境変数にクレデンシャル・キャッシュ・ディレクトリのロケーションを設定する。	CyberSafe のマニュアルを参照。クレデンシャル・キャッシュ・ディレクトリのデフォルト・ロケーションはプラットフォームごとに異なる。
ct_con_props を使用してクレデンシャル (必要なセキュリティ機能) を設定する、または ct_con_props を設定しないでデフォルト・クレデンシャルを使用する。	『Open Client Client-Library/C リファレンス・マニュアル』を参照。 セキュリティ機能がサポートされているかどうかを調べるには、ct_con_props と ct_config 内の CS_SUPPORTED アクション・タイプを使用する。
libtcl.cfg 設定ファイルのセキュリテイ・セクションを設定する。	『Open Client/Server 設定ガイド UNIX 版』を参照。
Sybase 提供の CyberSafe ドライバを、適切なフラグを使用して Client-Library アプリケーションにリンクする。	「Kerberos サポート・アプリケーションのコンパイルとリンク行」(10 ページ) を参照。
CyberSafe ユーティリティ kinit を使用して CyberSafe セキュリティ・メカニズムにログインし、Client-Library アプリケーションを実行する。	CyberSafe のマニュアルを参照。

コンパイルとリンク行

Client-Library と Server-Library は、Net-Library™、ディレクトリ・ドライバ、セキュリティ・ドライバに動的にリンクします。つまり、アプリケーションに次のオブジェクト・ファイルを明示的にリンクする必要はありません。

- Sybase Net-Library ドライバ (HP-UX と IBM RS/6000 の場合のリンカ・オプションは -linsck、Sun Solaris 2.x の場合は -ltli)
- Sybase ディレクトリ・ドライバ (リンカ・オプションは -lddce) またはセキュリティ・ドライバ (リンカ・オプションは -lsdce)

非スレッド・アプリケーション用のコンパイルとリンク行

以下の表は、UNIX が稼働し、Sybase がサポートするプラットフォーム上で、非スレッド Client-Library アプリケーションのコンパイルとリンクを行うコマンドの一般的なフォーマットを示します (コンパイルとリンクの情報については、*$SYBASE/$SYBASE_OCS/sample/ctlibrary* ディレクトリ内の *Makefile* も参照してください)。

表 1-4 は、静的ライブラリを使用して Client-Library アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 1-4: Client-Library の静的リンクとコンパイルのコマンド

プラットフォーム	コマンド
Sun Solaris 2.x	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib ¥ APP_FILES OCS_LIBS -Bstatic -lcs -ltcl -lcomn -lintl -Bdynamic -lnsl -ldl -lm -o program`
IBM RS/6000	`xlc_r4 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib APP_FILES OCS_LIBS ¥ -lcs -ltcl -lcomn -lintl -lm -o program`
HP 9000(8xx)	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib APP_FILES -Wl,a,archive ¥OCS_LIBS -lcs -ltcl -lcomn -lintl -Wl,-a,default -lcl -lm ¥ -lBSD -ldld -Wl,-E,+s -o program`
SGI	`cc -o [-n32 |-n64] -mips3 -I$SYBASE/$SYBASE_OCS/include` *`APP_FILES`* `¥ -L$SYBASE/$SYBASE_OCS/lib -Bstatic` *`OCS_LIBS`* `-lcs -ltd -lcomn ¥ -lintl -Bdynamic -lm -o program`
HP Tru64 UNIX	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib` *`APP_FILES OCS_LIBS`* `¥ -lcs -ltcl -lcomn -lintl -lm -o program`
Linux	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib` *`APP_FILES OCS_LIBS`* `-lcs -lsybtcl -lcomn -lintl -rdynamic -ldl -lnsl -lm -o program`

表 1-5 は、デバッグ・ライブラリを使用して Client-Library アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 1-5: Client-Library のデバッグ・リンクとコンパイルのコマンド

プラットフォーム	コマンド
Sun Solaris 2.x	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib ¥ -g` *`APP_FILES OCS_LIBS`* `-lcs -ltcl -lcomn -lintl -Bdynamic -lnsl -ldl -lm -o program`

プラットフォーム	コマンド
IBM RS/6000	`xlc_r4 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib -g APP_FILES ¥ OCS_LIBS -lcs -ltcl -lcomn -lintl -lm -o program`
HP 9000(8xx)	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib -g APP_FILES ¥ -Wl,-a,archive OCS_LIBS -lcs -ltcl -lcomn -lintl ¥-Wl,-a,default -lcl -lm -lBSD -ldld -Wl,-E,+s -o program`
SGI	`cc -g [-n32 |-n64] -mips3 -I$SYBASE/$SYBASE_OCS/include ¥ -L$SYBASE/$SYBASE_OCS/devlib APP_FILES OCS_LIBS -lcs -ltcl -linsck ¥ -lcomn -lintl -lm -o program`
HP Tru64 UNIX	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib APP_FILES OCS_LIBS ¥-lcs -ltcl -oldstyle_liblookup -lcomn -lintl -lm -o program`
Linux	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib APP_FILES OCS_LIBS -lcs -lsybtcl -lcomn -lintl -rdynamic -ldl -lnsl -lm -o program`

表 1-6 は、共有ライブラリを使用して Client-Library アプリケーションのコンパイルとリンク (動的ドライバを使用) を行うためのコマンドを示します。

表 1-6: Client-Library の共有リンクとコンパイルのコマンド

プラットフォーム	コマンド
Sun Solaris 2.x	`/opt/SUNWspro/bin/cI$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib ¥ -R$SYBASE/$SYBASE_OCS/lib APP_FILES OCS_LIBS -lcs -lcomn ¥ -ltcl -lintl -lnsl -ldl -lm -o program`
HP 9000(8xx)	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib APP_FILES OCS_LIBS -lcs -lcomn ¥-ltcl -lintl -linsck -Wl -lcl -lm -lBSD -o program`
SGI	`cc [-n32 |-n64] -mips3 -I$SYBASE/$SYBASE_OCS/include ¥ -L$SYBASE/$SYBASE_OCS/lib APP_FILES OCS_LIBS -lcs -ltcl -linsck ¥ -lcomn -lintl -lm -o program`

プラットフォーム	コマンド
HP Tru64 UNIX	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib` *`APP_FILES OCS_LIBS`* `-lcs -ltcl ¥ -oldstyle_liblookup -lcomn -lintl -lm -o program`

マルチスレッド・アプリケーションのコンパイルとリンク行

表 1-7 は、スレッドセーフ・サポートを利用するために、Client-Library アプリケーションをコンパイルしてライブラリとリンクするためのコマンドを示します。

表 1-7: Client-Library のスレッドセーフのリンクとコンパイルのコマンド

プラットフォーム	コマンド
Sun Solaris 2.3 以降	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/$SYNASE_OCS/include -L$SYBASE/$SYBASE_OCS/$SYBASE_OCS/lib -g ¥ -D_REENTRANT` *`APP_FILES OCS_LIBS`* `-lcs_r -ltcl_r -lcomn_r ¥-lintl_r -Bdynamic -lnsl -ldl -lpthread -lthread -lm -o program`
IBM RS/6000	`xlc_r4 -I$SYBASE/$SYBASE_OCS/$SYBASE/$SYBASE_OCS/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/$SYBASE/$SYBASE_OCS $SYBASE/$SYBASE_OCS/$SYBASE_OCS/lib -g -D_THREAD_SAFE ¥`*`APP_FILES OCS_LIBS`* `-lcs_r -ltcl_r -lcomn_r -lintl_r ¥ -lxdsxom -lpthread -lm -o program`
HP 9000(8xx)	`cc -I$SYBASE/$SYBASE_OCS/$SYBASE/$SYBASE_OCS/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/$SYBASE_OCS/lib -g -D_THREAD_SAFE ¥-D_REENTRANT -Ae` *`APP_FILES OCS_LIBS`* `-lcs_r ¥-ltcl_r -lcomn_r -lintl_r -Wl,-a,default -lcl -lm -lBSD ¥ -lpthread -lc_r -ldld -Wl,-E,+s -o program`
HP Tru64 UNIX	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -threads` *`APP_FILES ¥OCS_LIBS`* `-lcs_r -ltcl_r -lcomn_r -lintl_r -lm -o program`
Linux	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/vlib` *`APP_FILES OCS_LIBS`*`-lcs_r -lsybtcl_r -lcomn_r -lintl_r -rdynamic -ldl -lpthread -lnsl -lm -o program`

表 1-4 から表 1-7 に示したコンパイルとリンク行について次に説明します。

- *APP_FILES* は、アプリケーションのソース・ファイル (.c) またはオブジェクト・ファ イル (.o) を表します。
- *OCS_LIBS* は、コードが呼び出す Open Client と Open Server ライブラリをリンクするためのリンカ・オプションを表します。これらのオプションは、次に示すリンカ・オプションの一部または全部を次に示す順序で指定できます。
  - 非スレッド・アプリケーションの場合

    -lsrv (Server-Library ルーチン用)

    -lblk (Bulk-Library ルーチン用)

    -lct (Client-Library ルーチン用)
  - スレッド・アプリケーションの場合

    -lsrv_r (Server-Library ルーチン用)

    -lblk_r (Bulk-Library ルーチン用)

    -lct_r (Client-Library ルーチン用)

HP-UX システムのユーザの場合

- -W1、-a、archive オプションを使用すると、リンカは Sybase ライブラリを静的にリンクします。このオプションを指定していない場合、Client-Library は Sybase ライブラリの共有バージョンを使用します。共有ライブラリを使用する場合は、実行時に SH_LIB_PATH 環境変数に *$SYBASE/$SYBASE_OCS/lib* を設定しておく必要があります。また、アプリケーション・ユーザは *$SYBASE/$SYBASE_OCS/lib* 内のライブラリに対する read と execute のパーミッションが必要です。
- HP-UX は、アプリケーションが +s リンカ・オプションを使用してリンクされている場合を除いて、実行時に SH_LIB_PATH 環境変数を使用しません。システムが実行時に Sybase ライブラリを使用できるようにするには、+s リンカ・オプションを使用してください。-E は、実行時にドライバ・ライブラリがロードされるときに、未定義シンボル・エラーにならないようにするために必要です。詳細については、HP-UX の ld の man ページを参照してください。

SGI のユーザの場合

- 32 ビットのマシンに対しては -n32 オプションを使用し、64 ビットのマシンに対しては -n64 を使用します。

注意 環境変数 LD_LIBRARY_PATH に *$SYBASE/$SYBASE_OCS/lib* を設定して、共有 (動的) ライブラリにリンクされたプログラムを実行してください。デバッグ・モードでプログラムを実行するときは、LD_LIBRARY_PATH に *$SYBASE/$SYBASE_OCS/devlib* を設定してください。

HP Tru64 UNIX のユーザの場合

HP Tru64 UNIX 用のライブラリ・ファイルの拡張子は、*.a* (静的なライブラリ) と *.so* (共有ライブラリ) です。次の基本コマンドを使用して、Client-Library アプリケーションのコンパイルとリンクを行います。

非スレッド非リエントラント・ライブラリ

- 最適化ライブラリ ― 非スレッド、非リエントラント

```
cc -I$SYBASE/$SYBASE_OCS/include program.c ¥
 $SYBASE/$SYBASE_OCS/lib/libct.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcs.a ¥
 $SYBASE/$SYBASE_OCS/lib/libtcl.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcomn.a ¥
 $SYBASE/$SYBASE_OCS/lib/libintl.a ¥
 -lm -o program
```

- デバッグ・ライブラリ ― 非スレッド、非リエントラント

```
cc -g -I$SYBASE/$SYBASE_OCS/include ¥
 -L$SYBASE/$SYBASE_OCS/devlib program.c ¥
 -lct -lcs -ltcl -oldstyle_liblookup ¥
 -lcomn -lintl ¥
 -lm -o program
```

- 共有ライブラリ ― 非スレッド、非リエントラント

```
cc -I$SYBASE/$SYBASE_OCS/include
-L$SYBASE/$SYBASE_OCS/lib program.c ¥
 -lct -lcs -ltcl -oldstyle_liblookup ¥
 -lcomn -lintl ¥
 -lm -o program
```

スレッド・リエントラント・ライブラリ

- 最適化ライブラリ ― スレッド、リエントラント

```
cc -I$SYBASE/$SYBASE_OCS/include -threads program.c ¥
 $SYBASE/$SYBASE_OCS/lib/libct_r.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcs_r.a ¥
 $SYBASE/$SYBASE_OCS/lib/libtcl_r.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcomn_r.a ¥
 $SYBASE/$SYBASE_OCS/lib/libintl_r.a ¥
 -lm -o program
```

- デバッグ・ライブラリ ― スレッド、リエントラント

```
cc -I$SYBASE/$SYBASE_OCS/include -g -threads program.c ¥
 -L$SYBASE/$SYBASE_OCS/devlib ¥
 -oldstyle_liblookup -lct_r -lcs_r ¥
 -ltcl_r -lintl_r ¥
 -lm -o program
```

- 共有ライブラリ ― スレッド、リエントラント

```
cc -I$SYBASE/$SYBASE_OCS/include -threads program.c ¥
 -oldstyle_liblookup ¥
 -lct_r -lcs_r -ltcl_r ¥
 -lcomn_r -lintl_r ¥
 -lm -o program
```

Kerberos サポート・アプリケーションのコンパイルとリンク行

CyberSafe 用の Sybase ドライバは、動的にロードされる共有ライブラリです。CyberSafe セキュリティ・メカニズムを使用する Client-Library アプリケーションは適切なフラグを使用してリンクする必要があります。

CyberSafe が提供する Kerberos セキュリティ機能を使用して Client-Library アプリケーションをコンパイルする場合は、次のフラグを追加してください。

- Sun Solaris の場合

```
-u csfgss_pPtr -L/krb5/lib -lgss
```

- HP/UX の場合

```
-ldld -Wl, -E +s -Wl, -u, csfgss_pPtr
```

CyberSafe セキュリティを使用するアプリケーションを構築するには、CyberSafe アプリケーション・ツール・キット (*libgss.a*) が必要です。

バルク・コピー・ルーチン

バルク・コピー・ルーチンを使用する場合は、*libblk* バルク・コピー・ライブラリをリンクする必要があります。スレッド・アプリケーションでバルク・コピー・ルーチンを使用する場合は、*libblk_r* バルク・コピー・ライブラリをリンクします。

バルク・コピー・ライブラリをリンクするには次のようにします。

- 非スレッド・アプリケーションでは、リンク行の -lct の前に -lblk を追加します。
- マルチスレッド・アプリケーションでは、リンク行の -lct_r の前に -lblk_r を追加します。

バルク・コピーの詳細については、『Open Client/Server Common Libraries リファレンス・マニュアル』を参照してください。

パフォーマンスについて

共有ライブラリとリンクすると、静的ライブラリとリンクする場合よりも実行プログラムが小さくなり、リンク時間も少なくて済みます。ただし、共有ライブラリとリンクされた実行プログラムは、静的ライブラリを使用してリンクされた実行プログラムよりも起動に時間がかかります。さらに、静的ライブラリとは違って共有ライブラリは実行時に使用可能でなければなりません。

最高のパフォーマンスを提供するライブラリのタイプは、それぞれのサイトの稼働条件によって決まります。

ヘッダ・ファイル

ctpublic.h ヘッダ・ファイルは、すべての Client-Library アプリケーションのソース・ファイルにインクルードする必要があります。ほかの必要なヘッダ・ファイルは、*ctpublic.h* 内にネストされています。Bulk-Library を使用する場合は、*ctpublic.h* ではなく *bkpublic.h* をインクルードしてください。

ヘッダ・ファイルの詳細については、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。

Client-Library のサンプル・プログラムの使用

Client-Library には、Client-Library ルーチンの一般的な使い方の例を示すサンプル・プログラムがオンラインで提供されています。これらのサンプル・プログラムのコンパイルと実行には、次の情報を使用してください。

サンプル・プログラムには、Adaptive Server で提供されるサンプル・データベースを使用するものもあります。サンプル・データベースをインストールする方法の詳細については、インストール・ガイドを参照してください。必要なデータベースについては、各サンプル・プログラムの前提条件の説明を参照してください。

注意　Open Client 12.5 ワイド・テーブル機能と unichar 機能を示す新しいサンプル・プログラムが含まれています。これらのプログラムには、*uni_* または *wide_* プレフィクスが付いています。各プログラムについては、「サンプル・プログラムの概要」で説明しています。一部のサンプル・プログラムには、unipubs データベースが必要です。

---

makefile とサンプル・プログラム

すべてのプラットフォームでのサンプル・プログラムの構築に *makefile* を使用するには、使用しているコンパイラに合わせて SYBPLATFORM 環境変数を正しく設定してください。環境変数とライブラリ・パスの詳細については、表 B-1 (109 ページ) を参照してください。

共有ライブラリを使用するアプリケーションの構築

共有ライブラリを使用して IBM プラットフォームでアプリケーションを構築するには、次のようなリンク文を使用して makefile を設定します (次の文では、エクスポート・ファイルは **.exp* という名前です)。

```
xlc_r -g -D_THREAD_SAFE  -I. ¥
    -I$SYBASE/$SYBASE_OCS/include ¥
    -DDEBUG -Dnthread_rs6000=1 multthrd.c ¥
    thrdutil.o thrdfunc.o ¥
    -bI:$SYBASE/$SYBASE_OCS/libct_r.exp ¥
    -bI:$SYBASE/$SYBASE_OCS/libcs_r.exp ¥
    -bI:$SYBASE/$SYBASE_OCS/libcomn_r.exp ¥
    -lm -o multthrd
```

このようなアプリケーションを実行するには、次の条件を満たす必要があります。

- 共有ライブラリが、*$SYBASE/$SYBASE_OCS/lib* ディレクトリ内に格納されている。
- LIBPATH 環境変数が次のように設定されている。

  $SYBASE/$SYBASE_OCS/lib:/lib

- 設定ファイル *$SYBASE/$SYBASE_OCS/config/libtcl.cfg* が、適切なドライバを指定している。

サンプル・プログラムの目的

サンプル・プログラムは、Client-Library に固有な機能の例を示します。これらのプログラムは Client-Library のトレーニング用ではなく、アプリケーション・プログラマのためのガイドとして設計されています。サンプル・プログラムを使用する前に、各ソース・ファイルの先頭にある説明を読んで、ソース・コードの内容を確認してください。

これらの簡単なプログラムは、実際の運用環境で使用するために作成されているものではありません。実際の運用環境で使用できるレベルのプログラムでは、エラーや特殊なケースを処理するためのコードを追加する必要があります。

sybopts.sh スクリプトとアプリケーションの構築

sybopts.sh スクリプトはサンプル・プログラムに含まれています。このスクリプトによって、使用しているプラットフォームに対応する Open Client や Open Server アプリケーションを構築することができます。この場合、次のコマンドを実行して SYBPLATFORM 環境変数を読み込みます。

```
sybopts.sh <args>
```

args には次のいずれかの引数を指定します。

- compile － コンパイラのコマンドとプラットフォーム固有のコンパイル・フラグを返す。
- comlibs － アプリケーションにリンクさせなければならない必須の Sybase ライブラリのリストを返す。
- syslibs － アプリケーションにリンクさせなければならない Sybase 以外の必須ライブラリのリストを返す。

ロケーション

サンプル・プログラムは次のディレクトリに格納されています。

```
$SYBASE/sample/ctlibrary
```

このディレクトリには次のファイルが含まれています。

- サンプル・プログラムのオンライン・ソース・コード
- サンプル・プログラム用のデータ・ファイル
- サンプル・プログラムを構築するための *makefile*。*makefile* は、Client-Library アプリケーションの作成を開始するときに使用します。
- サンプル・ヘッダ・ファイル *example.h*
- サンプル・プログラムの構築、実行、テストの方法について説明している *README* ファイル

注意　*$SYBASE/sample/ctlibrary* の内容を、もとのファイルの整合性に影響を与えないでサンプル・プログラムを自由に使用できるような「作業」ディレクトリにコピーしてから、コンパイルして実行してください。

ヘッダ・ファイル

すべてのサンプル・プログラムはサンプル・ヘッダ・ファイル *example.h* を参照します。このファイルの内容は次のとおりです。

```
/*
 ** example.h
 **
 ** This is the header file that goes with the
 ** Sybase Client-Library example programs.
 **
 **
 */

 /*
 ** Define symbolic names, constants, and macros
 */
 #define EX_MAXSTRINGLEN     255
 #define EX_BUFSIZE          1024
 #define EX_CTLIB_VERSION    CS_VERSION_125
#define EX_BLK_VERSION    BLK_VERSION_125
#define EX_ERROR_OUT        stderr

 /*
 ** exit status values
 */
 #define  EX_EXIT_SUCCEED 0
 #define  EX_EXIT_FAIL 1

 /*
 ** Define global variables used in all sample
 ** programs
 */
 #define EX_SERVER           NULL/* use DSQUERY
                                    env var */
 #define EX_USERNAME         "user"
 #define EX_PASSWORD         "server_password"
```

サンプル・プログラムは、次のコード例に示すように *example.h* 内の define 文を使用します。

```
CS_CHAR *Ex_username = EX_USERNAME;
 CS_CHAR *Ex_password = EX_PASSWORD;

 /*
 ** If a user name is defined, set the
 ** CS_USERNAME property.
 */
 if (retcode == CS_SUCCEED && Ex_username != NULL)
 {
     if ((retcode = ct_con_props(*connection,
```

```
            CS_SET, CS_USERNAME, Ex_username,
            CS_NULLTERM, NULL)) != CS_SUCCEED)
        {
            ex_error("ct_con_props(username) failed");
        }
    }

    /*
    ** If a password is defined, set the
    ** CS_PASSWORD property.
    */
    if (retcode == CS_SUCCEED && Ex_password != NULL)
    {
        if ((retcode = ct_con_props(*connection,
            CS_SET, CS_PASSWORD, Ex_password,
            CS_NULLTERM, NULL)) != CS_SUCCEED)
        {
            ex_error("ct_con_props(password) failed");
        }
    }
```

example.h の中で編集する行について以下の項で説明します。

EX_USERNAME

EX_USERNAME は、*example.h* 内で “sa” と定義されています。サンプル・プログラムを実行する前に、*example.h* を編集して “sa” をサーバのログイン名に変更する必要があります。

EX_PASSWORD

EX_PASSWORD は、*example.h* 内で “ ” と定義されています。サンプル・プログラムを実行する前に、*example.h* を編集して “ ” をサーバのパスワードに変更できます。

EX_PASSWORD に関しては次のような 3 つのオプションがあります。必要に応じて適切なものを選択してください。

- オプション 1 － サンプルを実行している間だけ、サーバのパスワードを “server_password” に変更します。このオプションは、セキュリティを侵害される可能性があります。パスワードがこのような公開された値に設定されていると、承認されていないユーザでもサーバにログインできるからです。これでは問題がある場合は、次の 2 つの方法のどちらかを選択してください。
- オプション 2 － *example.h* 内の文字列 “server_password” を、使用するサーバのパスワードに変更します。オペレーティング・システムの保護メカニズムを使用して、使用中はほかのユーザがヘッダ・ファイルにアクセスできないようにします。サンプル・プログラムの使用を終了したら、変更した行を “server_password” に戻します。

- オプション 3 － サンプル・プログラム内で、サーバのパスワードを設定する ct_con_props コードを修正して、サンプルを使用するユーザにそのサーバのパスワードを要求するようにコードを変更します。このコードはプラットフォームに固有なので、Sybase からは提供されません。

サンプル・プログラムのためのユーティリティ・ルーチン

exutils.c ファイルには、ほかのすべての Client-Library サンプル・プログラムで使用されるユーティリティ・ルーチンが含まれています。この exutils.c は、アプリケーションが、より高いレベルのプログラムから Client-Library の実装の詳細部分を隠す方法を示しています。

これらのルーチンの詳細については、サンプル・ソース・ファイル内の先頭にあるコメントを参照してください。

サンプル・プログラムの概要

次のサンプル・プログラムがソフトウェアに含まれています。

uni_blktxt.c

サンプル・プログラム *uni_blktxt.c* は、バルク・コピー・ルーチンを使用して静的データをサーバ・テーブルにコピーします。プログラム変数にバインドされてサーバにまとめて送信される 3 つのローのデータがあります。このローは、テキスト・データを送信するために blk_textxfer を使用してもう一度送信されます。このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意 このサンプル・プログラムを実行するには、SQL Server バージョン 4.9.1 以降が必要です。

---

compute.c

サンプル・プログラム *compute.c* は、計算結果の処理の例を示すものです。このプログラムは、準備されたクエリを、言語コマンドを使用してサーバに送信します。標準の ct_results の while ループを使用して結果を処理し、カラム値をプログラム変数にバインドします。そのあと、標準の ct_fetch の while ループでローをフェッチして表示します。

準備されたクエリは次のとおりです。

```
select type, price from titles
 where type like "%cook"
 order by type, price
 compute sum(price) by type
 compute sum(price)
```

このクエリは、通常のローと計算ローの両方を返します。計算ローは 2 つの compute 句によって生成されます。最初の compute 句である “compute sum(price) by type” は、type の値が変化するたびに計算ローを生成します。2 つ目の compute 句である “compute sum(price)” は、最後に返される 1 つの計算ローを生成します。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意 このサンプル・プログラムを実行するには、pubs2 データベースが必要です。

---

uni_csr_disp.c

サンプル・プログラム *uni_csr_disp.c* は、読み込み専用カーソルの使い方を示します。このプログラムは、準備されたクエリでカーソルをオープンします。このプログラムは、標準の ct_results の while ループを使用して結果を処理し、カラム値をプログラム変数にバインドします。そのあと、標準の ct_fetch の while ループでローをフェッチして表示します。

準備されたクエリは次のとおりです。

```
select au_fname, au_lname, postalcode
 from authors
```

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意 このサンプル・プログラムを実行するには、SQL Server バージョン 10.0 以降と pubs2 データベースが必要です。

---

ex_alib.c と ex_amain.c

このサンプル･プログラムには *ex_alib.c* と *ex_amain.c* の 2 つのファイルがあり、Client-Library で、非同期ライトを行う方法を示します。このプログラムは、Client-Library によって提供される仕組みを使用して連続的なポーリングと、Client-Library の完了コールバックの使用を可能にします。

このサンプル・プログラムは次の 2 つのファイルで構成されます。

- *ex_alib.c* は、サンプル・プログラムのライブラリ部分のソース・コードを含んでいます。これは、非同期呼び出しを使用できるライブラリ・インタフェース部分です。このモジュールは、1 つの非同期オペレーション内でサーバにクエリを送信してサーバから結果を取り出す手段を提供します。
- *ex_amain.c* は、*ex_alib.c* によって提供されるサービスを使用するメイン・プログラムのソース・コードを含んでいます。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントと *EX_AREAD.ME* ファイルを参照してください。

exconfig.c

サンプル・プログラム *exconfig.c* は、Client-Library アプリケーションのプロパティを外部から設定する方法を示します。

このサンプル・プログラムを使用するには、Sybase インストール・ディレクトリ内にあるデフォルト・ランタイム設定ファイル *¥config¥ocs.cfg* を編集する必要があります。このサンプル・プログラムは、Client-Library プロパティ CS_CONFIG_BY_SERVERNAME を設定し、*server_name* パラメータに “server1” を指定して ct_connect を呼び出します。これに対して、Client-Library は外部設定ファイルで [Server1] セクションを探します。このサンプル・プログラムを実行するには、必要に応じて Sybase インストール・ディレクトリに *¥config¥ocs.cfg* を作成して、次のセクションを追加してください。

```
[server1]
 CS_SERVERNAME = real_server_name
```

real_server_name には、接続するサーバの名前を指定します。

Client-Library での外部設定ファイルの使用方法の詳細については、『Open Client Client-Library/C リファレンス・マニュアル』の「ランタイム設定ファイルの使い方」の項を参照してください。

firstapp.c

サンプル・プログラム *firstapp.c* は、サーバに接続し、select クエリを送信して、ローを表示する初歩的な例です。このサンプル・プログラムについては、『Open Client Client-Library/C プログラマーズ・ガイド』を参照してください。

getsend.c

サンプル・プログラム *getsend.c* は、テキストとその他のデータ型を含んでいるテーブルから、text データを取得して更新する方法を示します。この例に示す処理は、image データの検索と更新に使用することもできます。このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意　このサンプル・プログラムを実行するには、SQL Server バージョン 10.0 以降が必要です。

i18n.c

i18n.c サンプル・プログラムは、Client-Library で使用できる次のような国際化機能の一部を示します。

- ローカライズされたエラー・メッセージ
- ユーザ定義のバインド・タイプ

このプログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

multthrd.c と *thrdfunc.c*

このサンプル・プログラムには、マルチスレッド Client-Library アプリケーションの例を示す *multthrd.c* と *thrdfunc.c* という 2 つのファイルが含まれています。この 2 つのファイルには、次のような情報が含まれています。

- *multthrd.c* は、5 つのスレッドを生成するソース・コードを含んでいます。各スレッドは 1 つのカーソルまたは 1 つの通常のクエリを処理します。メイン・スレッドはほかのスレッドがクエリ処理を完了するまで待ってから終了します。
- *thrdfunc.c* は、サンプル・プログラムがプラットフォームに応じて実行に使用するスレッド・ルーチンと同期化ルーチンを決定するプラットフォーム固有の情報を含んでいます。

このプログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

このサンプル・プログラムは Client-Library によってサポートされるスレッド・パッケージがプラットフォーム上に存在しない場合は実行できません。*multthrd.c* サンプル・プログラムは DCE pthread API を使用します。*multthrd.c* サンプル・プログラムを構築するには、使用するマシンに DCE をインストールして稼働させる必要があります。また、「付録 B 環境変数」の説明に従って SYBPLATFORM 環境変数を設定する必要もあります。

注意　このサンプル・プログラムを実行するには、SQL Server バージョン 10.0 以降が必要です。

rpc.c

RPC コマンドのサンプル・プログラム *rpc.c* は、RPC コマンドをサーバに送信してその結果を処理します。このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意 このサンプル・プログラムを実行するには、SQL Server バージョン 4.9.1 以降が必要です。

---

secct.c

サンプル・プログラム *secct.c* は、Client-Library アプリケーションでネットワーク・ベースのセキュリティ機能を使用する方法を示します。

このサンプル・プログラムを実行するには、使用するマシンに DCE または CyberSafe Kerberos をインストールして稼働させる必要があります。また、Security Guardian や Open Server のサンプル・プログラム *secsrv.c* などの、ネットワーク・ベースのセキュリティをサポートするサーバに接続することも必要です。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。ネットワーク・セキュリティ・サービスの詳細については、『Open Client/Server 設定ガイド UNIX 版』を参照してください。

uni_blktxt.c

サンプル・プログラム *uni-blktxt.c* は、バルク・コピー・ルーチンを使用して静的データをサーバ・テーブルにコピーします。このプログラムは、unichar データ型と univarchar データ型を使用するために修正されています。プログラム変数にバインドされてサーバにまとめて送信される 3 つのローのデータがあります。このローは、テキスト・データを送信するために blk_textxfer を使用してもう一度送信されます。

uni_compute.c

サンプル・プログラム *uni_compute.c* は、計算結果の処理の例を示すものです。このサンプル・プログラムは unichar データ型と univarchar データ型を使用するために *compute.c* サンプル・プログラムを修正したものです。実行するには unipubs データベースが必要です。このプログラムは、準備されたクエリを、言語コマンドを使用してサーバに送信します。このプログラムは、標準の ct_results ループを使用して結果を処理し、カラム値をプログラム変数にバインドします。そのあと、標準の ct_fetch ループでローをフェッチして表示します。

uni_csr.c

サンプル・プログラム *uni_csr_disp.c* は、読み込み専用カーソルの使い方を示します。このサンプル・プログラムは *csr_disp.c* サンプル・プログラムを修正したものです。実行するには unipubs データベースが必要です。このプログラムは、準備されたクエリでカーソルをオープンします。このプログラムは、標準の ct_results の while ループを使用して結果を処理し、カラム値をプログラム変数にバインドします。そのあと、標準の ct_fetch の while ループでローをフェッチして表示します。

準備されたクエリは次のとおりです。

```
select au_fname, au_lname, postalcode
 from authors
```

uni_firstapp.c

このプログラムは、unichar データ型と univarchar データ型を使用するように *firstapp.c* サンプル・プログラムを修正したものです。これは、サーバに接続し、select クエリを送信して、ローを表示する初歩的な例です。*firstapp.c* プログラムについては、『Open Client Client-Library/C プログラマーズ・ガイド』を参照してください。

uni_rpc.c

RPC コマンドのサンプル・プログラム *uni_rpc.c* は、RPC コマンドをサーバに送信してその結果を処理します。このサンプル・プログラムは unichar データ型と univarchar データ型を使用するために *rpc.c* サンプル・プログラムを修正したものです。実行するには unipubs データベースが必要です。

このプログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

usedir.c

このサンプル・プログラムは、使用可能なサーバのリストをディレクトリ・サービスに問い合わせる Client-Library の機能を示します。

usedir.c は、ドライバ設定ファイル内の定義に従ってデフォルト・ディレクトリで Sybase サーバ・エントリを検索します。ネットワーク・ディレクトリ・サービスが使用されていない場合、*usedir.c* は *interfaces* ファイルにサーバ・エントリがあるかどうかを調べます。そのあと、検索された各エントリの内容を表示して、接続するサーバをユーザが選択できるようにします。

このプログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。ネットワーク・ディレクトリ・サービスの詳細については、『Open Client/Server 設定ガイド UNIX 版』を参照してください。

wide_*compute.c*

サンプル・プログラム *wide_compute.c* は、ワイド・テーブルと大きなカラム・サイズによる計算結果の処理を示します。このサンプル・プログラムは Open Client と Open Server のバージョン 12.5 に実装されています。このプログラムは、準備されたクエリを、言語コマンドを使用してサーバに送信します。このプログラムは、標準の ct_results の while ループを使用して結果を処理し、カラム値をプログラム変数にバインドします。そのあと、標準の ct_fetch の while ループでローをフェッチして表示します。

準備されたクエリは次のとおりです。

```
select type, price from titles
 where type like "%cook"
 order by type, price
 compute sum(price) by type
 compute sum(price)
```

このクエリは、通常のローと計算ローの両方を返します。計算ローは 2 つの compute 句によって生成されます。最初の compute 句である “compute sum(price) by type” は、type の値が変化するたびに計算ローを生成します。2 つ目の compute 句である “compute sum(price)” は、最後に返される 1 つの計算ローを生成します。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意 このサンプル・プログラムを実行するには、pubs2 データベースが必要です。

---

wide_curupd.c

このプログラムは、カーソルを使用して pubs2 データベース内の “publishers” というテーブルからデータを取り出します。ローごとにデータを取得し、publishers テーブル内の state カラムに新しい値を入力するプロンプトを表示します。

更新のために、入力パラメータ用の値 (publishers テーブルの state カラム) を入力します。次に示すコマンドを実行して publishers3 テーブルを作成してから、サンプル・プログラムを実行してください。

```
use pubs2

go

drop table publishers3

go

create table publishers3 (pub_id char(4) not null,

pub_name varchar(400) null, city varchar(20) null,
```

```
state char(2) null)
go
select * into publishers3 from publishers
go
create unique index pubind on publishers3(pub_id)
```

wide_dynamic.c

このプログラムは、カーソルを使用して pubs2 データベース内の “publishers” というテーブルからデータを取り出します。ローごとにデータを取得し、publishers テーブル内の “state” というカラムに新しい値を入力するプロンプトを表示します。

このプログラムは、動的 SQL を使用して tempdb データベース内の titles テーブルから値を取り出します。識別子の付いたプレースホルダを含む select 文が、サーバに送信されて部分的にコンパイルされ、保存されます。したがって、select を呼び出すたびに、取得されるローを決定するキー値の新しい値だけを渡します。動作は、ストアド・プロシージャに入力パラメータを渡す動作に似ています。また、このプログラムはカーソルを使用してローを 1 つずつ取得します。必要に応じて、この操作を実行できます。

wide_rpc.c

RPC コマンドのサンプル・プログラム *rpc.c* は、RPC コマンドをサーバに送信してその結果を処理します。この動作は *rpc.c* プログラムと同じですが、異なる点は、ワイド・テーブルと大きなカラム・サイズを使用することです。

このプログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

第 2 章

Open Client DB-Library/C

Open Client DB-Library はクライアント・アプリケーションの作成に使用できるルーチンの集まりです。DB-Library は、Client-Library 以前の古いルーチンです。ディレクトリ・サービスやセキュリティ・サービスのサポートなどの新しい機能は DB-Library には含まれていません。これらの新しいサービスを利用する場合は Client-Library を使用してください。

DB-Library には、サーバにコマンドを送信するルーチンとそれらのコマンドの結果を処理するルーチンが含まれています。アプリケーション・プロパティの設定、エラー条件の処理、サーバとのアプリケーションの対話に関するさまざまな情報の提供を行うルーチンもあります。

この章の内容は、次のとおりです。

一般的な条件

サンプル・プログラムをはじめとする DB-Library アプリケーションを実行するには、以下の準備が必要です。

- 次の環境変数を設定します。詳細については、「付録 B 環境変数」を参照してください。
  - SYBASE
  - DSQUERY
  - SYBPLATFORM
  - プラットフォーム固有のライブラリ・パス変数
- Adaptive Server に接続できるようにします。Adaptive Server に接続する方法については、『Open Client/Server 設定ガイド UNIX 版』を参照してください。
- *$SYBASE/sample* 下の各製品のディレクトリにある *README* ファイルを読みます。サンプル・プログラムの実行方法の詳細な手順については、*README* ファイルを参照してください。

それぞれのサンプル・プログラムの説明およびその他の前提条件については、次の項を参照してください。

DB-Library 実行プログラムの構築

この項では、ライブラリ、リンクの方法、ヘッダ・ファイルについて説明します。

ライブラリ

次の表は、すべての DB-Library 機能を十分に活用するために組み込む必要のあるライブラリを示します。表の 1 行目には、すべてのプラットフォームで使用できるライブラリを示します。2 行目以降は各プラットフォームに固有なライブラリを示します。

表 2-1: プラットフォーム固有のライブラリ

プラットフォーム	必要なライブラリ
すべてのプラットフォーム	*libsybdb* − DB-Library (Sybase) *libm* − UNIX 標準の算術ライブラリ (システム)
Sun Solaris 2.x	*libnsl* − ネットワーク・ライブラリ (システム)
HP 9000(8xx)	*libcl* − トランスポート制御層 (システム) *libBSD* − BSD ライブラリ (システム)

リンクとコンパイル行

以下の表は、UNIX オペレーティング・システムが稼働し、Sybase がサポートするプラットフォーム上で、DB-Library アプリケーションのコンパイルとリンクを行うコマンドの一般的なフォーマットを示します。表 2-2 は、静的ライブラリを使用して DB-Library アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 2-2: DB-Library の静的リンクとコンパイルのコマンド

プラットフォーム	コマンド
Sun Solaris 2.x	`cc -I$SYBASE/include -L$SYBASE/lib program.c -Bstatic -lsybdb ¥` `-lnsl -Bdynamic -lm -o program`
IBM RS/6000	`xlc_r4 -I$SYBASE/include -L$SYBASE/lib program.c -lsybdb -lm ¥` `-o program`
HP 9000(8xx)	`cc -I$SYBASE/include -L$SYBASE/lib program.c -lsybdb ¥` `-Wl,-a,archive -lcl -lm -lBSD -o program`

プラットフォーム	コマンド
SGI	`cc [-n32 | -n64] -mips3 -I$SYBASE/include program.c ¥` `$SYBASE/lib/libsybdb.a -lm -o program`
HP Tru64 UNIX	`cc -I$SYBASE/include program.c ¥` `$SYBASE/lib/libsybdb.a -ldnet -lm -o program`

表 2-3 は、デバッグ・ライブラリを使用して DB-Library アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 2-3: DB-Library のリンクとコンパイルのコマンド

プラットフォーム	コマンド
Sun Solaris 2.x	`cc -I$SYBASE/include -L$SYBASE/devlib program.c -lsybdb -lnsl ¥` `-lm -o program`
IBM RS/6000	`xlc_r4 -I$SYBASE/include -L$SYBASE/devlib program.c -lsybdb -lm ¥` `-o program`
HP 9000(8xx)	`cc -I$SYBASE/include -L$SYBASE/devlib program.c -lsybdb ¥` `-linsck -Wl,-a, archive -lcl -lm -lBSD -o program`
SGI	`cc -g [-n32 | -n64] -mips3 -I$SYBASE/include -L$SYBASE/devlib ¥` `program.c -lsybdb -lm -o program`
HP Tru64 UNIX	`cc -g -I$SYBASE/include -L$SYBASE/devlib program.c ¥` `-lsybdb -ldnet -lm -o program`

表 2-4 は、共有ライブラリをサポートするプラットフォーム上で DB-Library アプリケーションのコンパイルとリンク (動的ドライバを使用) を行うためのコマンドを示します。

表 2-4: DB-Library の共有コンパイルとリンクのコマンド

プラットフォーム	コマンド
Sun Solaris 2.x	`cc -I$SYBASE/include -L$SYBASE/lib -R$SYBASE/lib program.c ¥` `-lsybdb -lnsl -lm -o program`
HP 9000(8xx)	`cc -I$SYBASE/include -L$SYBASE/lib program.c -lsybdb -Wl -lcl ¥` `-lm -lBSD -o program`
SGI	`cc [-n32 | -n64] -mips3 -I$SYBASE/include -L$SYBASE/lib ¥` `program.c -lsybdb -lm -o program`
Digital UNIX	`cc -I$SYBASE/include -L$SYBASE/lib ¥` `program.c -lsybdb -ldnet -lm -o program`

プラットフォーム	コマンド
HP Tru64 UNIX	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib ¥ program.c -lsybdb -ldnet ¥ -lm -o program`

パフォーマンスについて

共有ライブラリとリンクすると、静的ライブラリとリンクする場合よりも実行プログラムが小さくなり、リンク時間も少なくて済みます。ただし、共有ライブラリとリンクされた実行プログラムは、静的ライブラリを使用してリンクされた実行プログラムよりも起動に時間がかかります。さらに、静的ライブラリとは違って共有ライブラリは実行時に使用可能でなければなりません。

最高のパフォーマンスを提供するライブラリのタイプは、それぞれのサイトの稼働条件によって決まります。

ヘッダ・ファイル

DB-Library/C アプリケーションはすべて、次のヘッダ・ファイルを必要とします。

- *sybfront.h* － 関数の戻り値 (『Open Client DB-Library/C リファレンス・マニュアル』参照) や、終了値 STDEXIT と ERREXIT などの記号定数を定義しています。*sybfront.h* ファイルには、プログラム変数の宣言に使用できるデータ型の型定義も含まれています。
- *sybdb.h* － 定義と型定義が追加されています。これらの定義のほとんどは DB-Library/C ルーチンだけが使用します。*sybdb.h* の内容は、『Open Client DB-Library/C リファレンス・マニュアル』の説明に従って使用してください。
- *syberror.h* － エラー重大度の値を含んでいます。プログラムがこれらの値を参照する場合はインクルードする必要があります。

ヘッダ・ファイルの詳細については、『Open Client DB-Library/C リファレンス・マニュアル』を参照してください。

DB-Library サンプル・プログラムの使用

DB-Library には、DB-Library ルーチンの一般的な使い方を示すサンプル・プログラムがオンラインで提供されています。

サンプル・プログラムには、Adaptive Server で提供されるサンプル・データベースを使用するものもあります。サンプル・データベースをインストールする方法の詳細については、インストール・ガイドを参照してください。

サンプル・プログラムの目的

System 11 のサンプル・プログラムは、DB-Library/C に固有の機能の例を示します。これらのプログラムは DB-Library のトレーニング用ではなく、アプリケーション・プログラマのためのガイドとして設計されています。サンプル・プログラムを使用する前に、各ソース・ファイルの先頭にある説明を読んで、ソース・コードの内容を確認してください。

注意　これらの簡単なプログラムは、実際の運用環境で使用するために作成されているものではありません。実際の運用環境で使用できるレベルのプログラムでは、エラーや特殊なケースを処理するためのコードを追加する必要があります。

ロケーション

サンプル・プログラムは次のディレクトリに格納されています。

```
$SYBASE/sample/dblibrary
```

このディレクトリには次のようなファイルが含まれています。

- サンプル・プログラムのオンライン・ソース・コード
- サンプル・プログラム用のデータ・ファイル
- サンプル・ヘッダ・ファイル *sybdbex.h*
- サンプル・プログラムの構築、実行、テストの方法について説明している *README* ファイル

注意　*$SYBASE/sample/dblibrary* の内容を、もとのファイルの整合性に影響を与えないでサンプル・プログラムを自由に使用できるような「作業」ディレクトリにコピーしてから、コンパイルして実行してください。

ヘッダ・ファイル

すべてのサンプル・プログラムは、サンプル・ヘッダ・ファイル *sybdbex.h* を参照します。*sybdbex.h* の内容は、次のとおりです。

```
/*
 ** sybdbex.h
 **
 ** This is the header file that goes with the
 ** Sybase DB-Library example programs.
 **
 **
 */

 #define USER              "user"
 #define PASSWORD          "server_password"
 #define LANGUAGE          "us_english"
 #define SQLBUFLEN         255
 #define ERR_CH            stderr
 #define OUT_CH            stdout
 extern void               error();
 extern int                err_handler();
 extern int                msg_handler();
```

例 5 以外のすべてのサンプル・プログラムには、次の行が含まれています。

```
DBSETLUSER(login, USER);
DBSETLPWD(login, PASSWORD);
```

sybdbex.h の中で編集する行について次に説明します。

- USER は、*sybdbex.h* 内で “user” と定義されています。サンプル・プログラムを実行する前に、*sybdbex.h* を編集して “user” をサーバのログイン名に変更しておかなければなりません。
- PASSWORD は、*sybdbex.h* 内で “server_password” と定義されています。サンプル・プログラムを実行する前に、*sybdbex.h* を編集して “server_password” をサーバのパスワードに変更しておかなければなりません。PASSWORD について次のいずれかのオプションを選択してください。

  オプション 1：サンプル・プログラムを実行している間だけ、サーバ・パスワードを “server_password” に変更します。このオプションは、セキュリティを侵害される可能性があります。パスワードがこのような公開された値に設定されていると、承認されていないユーザでもサーバにログインできるからです。このような場合は、次のいずれかを行います。

  オプション 2：sybdbex.h 内の文字列 “server_password” を、使用するサーバのパスワードに変更します。オペレーティング・システムの保護メカニズムを使用して、使用中はほかのユーザがヘッダ・ファイルにアクセスできないようにします。サンプル・プログラムの使用を終了したら、変更した行を “server_password” に戻します。

オプション 3：サンプル・プログラム内で、DBSETLPWD 行全体を削除して、ユーザにそのサーバのパスワードを要求するようにコードを変更します。このコードはプラットフォームに固有なので、Sybase からは提供されません。

- LANGUAGE：使用するサーバの言語が英語でない場合は、*sybdbex.h* を編集して LANGUAGE 行にサーバと同じ言語を指定します。LANGUAGE を参照するサンプル・プログラムは例 12 のみです。

サンプル・プログラムの概要

次のようなサンプル・プログラムがソフトウェアに含まれています。

例 1：クエリの送信、バインド、結果の表示

example1.c サンプル・プログラムは、1 つのコマンド・バッチで 2 つのクエリを Adaptive Server に送信し、結果をバインドして、返されたデータのローを出力します。

例 2：新しいテーブルへのデータ挿入

example2.c は、新しく作成されたテーブルにファイルからデータを挿入し、サーバのローを選択して、結果のバインドと出力を行います。このサンプルを使用するには、提供されている *datafile* というファイルが必要です。また、ログイン・データベース内で create database パーミッションを持っていることを前提とします。

例 3：集約結果と計算結果のバインド

example3.c サンプル・プログラムは、pubs2 データベース内の titles テーブルから情報を選択して出力します。サンプル・プログラムは、集約結果と計算結果の両方のバインドの例を示すものです。

注意　このサンプル・プログラムを実行するには、Adaptive Server と pubs2 データベースにアクセスする必要があります。

例 4：ロー・バッファリング

example4.c サンプル・プログラムはロー・バッファリングの例です。このプログラムは、Adaptive Server にクエリを送信し、返されたローをバッファに入れて、それらを対話的に調べることができるようにします。

例 5：データ変換

example5.c サンプル・プログラムは、データ変換を処理する DB-Library/C ルーチン dbconvert の例を示します。

例 6：ブラウズ・モードでの更新

example6.c サンプル・プログラムは、ブラウズ・モードについての例です。このサンプル・プログラムはテーブルを作成して、そのテーブルにデータを挿入し、ブラウズ・モード・ルーチンを使用してそのテーブルを更新します。ブラウズ・モードはデータのローを一度に 1 つずつ更新する必要があるアプリケーションに便利です。

注意 *example6.c* を使用するには、提供されている *datafile* というファイルが必要です。このプログラムはデフォルトのデータベース内に *alltypes* というテーブルを作成します。

例 7：ブラウズ・モードとアドホック・クエリ

example7.c サンプルは、ブラウズ・モードを使用してアドホック・クエリによる結果カラムのソースを調べます。ブラウズ可能なテーブルから導出されたカラム、および SQL 式の結果ではないカラムを更新できるのはブラウズ・モードのアプリケーションだけなので、結果カラムのソースを調べることは重要です。

このサンプル・プログラムは、ブラウズ・モードを使用して更新できる、アドホック・クエリによる結果カラムはどれであるかをアプリケーションが調べる方法を示します。また、このサンプル・プログラムは、アドホック・クエリの入力を要求します。select クエリがキーワード for browse を含んでいるかどうか、選択されるテーブルをブラウズできるかどうかによって、どのように結果が異なるかに注目してください。

例 8：RPC (リモート・プロシージャ・コール) の実行

example8.c サンプル・プログラムは、RPC (リモート・プロシージャ・コール) を送信し、その RPC による結果ローを表示して、リモート・プロシージャによって返されたパラメータとステータスを表示します。

このサンプル・プログラムでは、デフォルト・データベース内にストアド・プロシージャ rpctest を作成しておかなければなりません。*example8.c* ソース・コードの先頭のコメントは、rpctest を作成するのに必要な create procedure 文を指定します。

例 9：text ルーチンと image ルーチン

example9.c サンプル・プログラムは、次の手順に従って、ランダムなイメージを生成してテーブルに挿入し、そのイメージを選択してもとのイメージと比較する例を示します。

1　text または image 値以外のすべてのデータをローに挿入 (insert) します。

2　ローを更新 (update) して text 値または image 値を NULL に設定します。これは必須の手順です。null 値を含む text または image カラム・ローに有効なテキスト・ポインタが割り当てられるのは、その null 値が update 文によって明示的に入力された場合に限られるからです。

3　ローを選択 (select) します。text 値または image 値を含むカラムを明示的に選択 (select) します。これは、アプリケーションの DBPROCESS に正しいテキスト・ポインタとテキスト・タイムスタンプ情報を提供するために必要な手順です。アプリケーションは、この select によって返されたデータを破棄します。

4　dbtxtptr を呼び出して、DBPROCESS からテキスト・ポインタを取り出します。dbtxtptr の *column* パラメータは整数であり、手順 3 で実行した select を参照します。たとえば、次のような select を実行したとします。

```
select date_column, integer_column, text_column
     from bigtable
```

ここでは、text_column は text カラムの名前です。dbtxtptr は、*column* パラメータが 3 として渡されるように要求します。

5　dbtxtimestamp を呼び出して、DBPROCESS からテキスト・タイムスタンプを取り出します。dbtxtimestamp の column パラメータは、手順 3 で実行した select を参照します。

6　text 値または image 値を Adaptive Server に書き込みます。アプリケーションは次のどちらかを実行できます。

- 1 回の dbwritetext 呼び出しで値を書き込む。
- dbwritetext と dbmoretext を使用して、まとまりごとに値を書き込む。

7　アプリケーションに、この text 値または image 値に対してさらに更新を実行させるときは、正常に実行された dbwritetext のオペレーションの結果として Adaptive Server によって返される新しいテキスト・タイムスタンプを保存しなければならない場合があります。新しいテキスト・タイムスタンプには、dbtxtsnewval を使用してアクセスします。また、あとで取り出せるように dbtxtsput を使用して保存できます。

注意　このサンプル・プログラムを実行するには、pubs2 データベースを格納している Adaptive Server にアクセスする必要があります。

例 10：image の挿入

example10.c サンプル・プログラムは、作家 ID と image を含んでいるファイルの名前を要求し、そのファイルから image を読み込んで、作家 ID と image を含んでいる新しいローを pubs2 データベースの “au_pix” というテーブルに挿入します。text 値または image 値をデータベース・テーブルに挿入する方法については、例 9 を参照してください。

注意 このサンプル・プログラムを実行するには、pubs2 データベースを格納している Adaptive Server にアクセスする必要があります。作家 ID の形式は “000-00-0000” でなければなりません。サンプル・コードと一緒に提供される *imagefile* ファイルには image が含まれています。

例 11：image の検索

example11.c サンプル・プログラムは、pubs2 データベース内の au_pix テーブルから image を取得します。入力する作家 ID によって、プログラムが選択するローが決まります。ローを検索したあと、このサンプル・プログラムは pic カラムに含まれている image を指定のファイルにコピーします。

Adaptive Server から text または image 値を取得するには 2 つの方法があります。

- このサンプル・プログラムでは、値を含んでいるローを選択し、dbnextrow を使用してそのローを処理します。dbnextrow が呼び出されると、dbdata を使用して、返されたイメージへのポインタを返すことができます。
- そのほかに、dbmoretext と一緒に dbreadtext を使用して、text または image 値をさらに小さないくつかのデータのまとまりとして読み込むこともできます。

dbreadtext の詳細については、『Open Client DB-Library/C リファレンス・マニュアル』を参照してください。

注意 このサンプル・プログラムを実行するには、Adaptive Server と pubs2 データベースにアクセスする必要があります。

例 12：国際言語ルーチン

example12.c サンプル・プログラムは、pubs2 データベースからデータを取得して us_english フォーマットで出力します。

注意 このサンプル・プログラムを実行するには、Adaptive Server と pubs2 データベースにアクセスする必要があります。

例 13：バルク・コピー

バルク・コピーのサンプル・プログラム *bulkcopy.c* は、バルク・コピー・ルーチンを使用して、Adaptive Server の数種のデータ型を含んでいる新しく作成されたテーブルに、ホスト・ファイルからデータをコピーします。

注意　このサンプル・プログラムを実行するには、Adaptive Server にアクセスする必要があります。create database と create table パーミッションを持っていることも必要です。

例 14：2 フェーズ・コミット

2 フェーズ commit のサンプル・プログラム *twophase.c* は、2 つの異なるサーバに対して簡単な更新を実行します。実際の更新内容については、ソース・コードを参照してください。このサンプル・プログラムを実行したあとは、各サーバに対して isql を使用して、更新が実際に行われたかどうかを調べることができます。

このサンプル・プログラムでは、SERVICE と PRACTICE という名前の 2 つのサーバ上で Adaptive Server が稼働していて、各サーバに pubs2 データベースが格納されていなければなりません。使用するサーバがこれとは違った名前である場合は、ソース・コード内の SERVICE と PRACTICE を実際に使用するサーバの名前で置き換えてください。

このサンプル・プログラムを実行する前に、クライアントが両方のサーバにアクセスできることを確認しておく必要があります。Adaptive Server の複数インスタンスに接続する方法については、『Open Client/Server 設定ガイド UNIX 版』を参照してください。

注意　PRACTICE サーバが SERVICE サーバとは別のマシン上にある場合は、PRACTICE サーバを SERVICE クエリ・ポートに接続できなければなりません。詳細については、『Open Client/Server **設定ガイド UNIX 版**』を参照してください。

第 3 章

Open Server Server-Library/C

Open Server Server-Library/C は、クライアント／サーバ・アーキテクチャの機能を利用するサーバを設計するために使用します。これらの Open Server は、外部データベース管理システムに保管されているデータにアクセスし、外部イベントをトリガし、Open Client アプリケーションに応答できます。

クライアント／サーバ・アーキテクチャでは、コンピューティング作業が「クライアント」と「サーバ」間で分担されます。

- クライアントはサーバに要求し、サーバの応答を処理します。
- サーバは要求に応えて、データ、パラメータ、ステータス情報をクライアントに返します。

このアーキテクチャでは、Open Client アプリケーション・プログラムは、Adaptive Server と Open Server によって提供されるサービスを使用するクライアントになります。Server-Library を使用すると、完全なスタンドアロン・サーバを作成できます。

この章の内容は、次のとおりです。

一般的な条件

サンプル・プログラムをはじめとする Open Server アプリケーションを実行するには、以下の準備が必要です。

- Adaptive Server と pubs2 サンプル・データベースにアクセスできる必要があります。pubs2 データベースをインストールする方法については、インストール・ガイドを参照してください。

- 次の環境変数を設定します。詳細については、「付録 B 環境変数」を参照してください。
  - SYBASE
  - DSQUERY と DSLISTEN
  - SYBPLATFORM
  - プラットフォーム固有のライブラリ・パス変数
- Adaptive Server に接続できるようにします。Adaptive Server に接続する方法については、『Open Client/Server 設定ガイド UNIX 版』を参照してください。

Server-Library 実行プログラムの構築

この項では、ライブラリ、リンクの方法、ヘッダ・ファイルについて説明します。

ライブラリ

次の表は、すべての Server-Library 機能を十分に活用するために組み込む必要があるライブラリを示します。表の 1 行目には、すべてのプラットフォームで使用するライブラリを示します。2 行目以降はプラットフォームに固有なライブラリを示しています。

表 3-1: プラットフォーム固有のライブラリ

プラットフォーム	必要なライブラリ
すべてのプラットフォーム	*libct*> − Client-Library (Sybase) *libcs*> − CS-Library (Sybase) *libtcl* − トランスポート制御層 (Sybase 内部使用) *libcomn* − 内部共有ユーティリティ・ライブラリ (Sybase 内部使用) *libintl* − 国際化サポート・ライブラリ (Sybase 内部使用) *libsrv* − Server-Library (Sybase) *libsybdb* − DB-Library (Sybase)

コンパイルとリンク行のコマンド

以下の表は、UNIX オペレーティング・システムが稼働し、Sybase がサポートするプラットフォーム上で、Server-Library アプリケーションのコンパイルとリンクを行うコマンドの一般的なフォーマットを示します。以下の 3 つの表はそれぞれコンパイルとリンク行のコマンドを示します。

表 3-2 は、静的ライブラリを使用して Server-Library アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 3-2: Server-Library の静的リンクとコンパイルのコマンド

プラットフォーム	コマンド

Sun
Solaris 2.x
	`/opt/SUNWspro/bin/cc -I$SYBASE/include -L$SYBASE/lib program.c ¥`
	`-Bstatic -lsrv [-lsybdb | -lct] -lcs -ltcl -lcomn -lintl ¥`
	`-Bdynamic -lnsl -lm -o program`

IBM RS/6000	`xlc_r4 -I$SYBASE/include -L$SYBASE/lib program.c -lsrv ¥` `[-lsybdb| -lct] -lcs -lcomn -ltcl -lintl -lm -o program`

HP 9000(8xx)	`cc -I$SYBASE/include -L$SYBASE/lib program.c -Wl,-a,archive ¥`
	`-lsrv [-lsybdb | -lct] -lcs -lcomn -ltcl -lintl -linsck ¥`
	`-lcl -lm -lBSD -o program`

SGI	`cc -o [-n32 | -n64] -mips3 -I$SYBASE/include -L$SYBASE/lib ¥`
	`program.c -lsrv -Bstatic [-lsybdb | -lct] ¥`
	`-lcs -ltcl -lcomn -lintl -Bdynamic -lm -o progam`

HP Tru64 UNIX	`cc -I$SYBASE/$SYBASE_OCS/include program.c¥`
	`$SYBASE/$SYBASE_OCS/lib/libsrv.a ¥`
	`[$SYBASE/$SYBASE_OCS/lib/libsybdb.a | $SYBASE/$SYBASE_OCS/lib/libct.a] ¥`
	`$SYBASE/$SYBASE_OCS/lib/libcs.a $SYBASE/$SYBASE_OCS/lib/libtcl.a ¥`
	`$SYBASE/$SYBASE_OCS/lib/libcomn.a ¥`
	`$SYBASE/$SYBASE_OCS/lib/libintl.a ¥`
	`-lm -o program`

Linux	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsrv [-lsybdb|-lct] -lcs -lsybtcl -lcomn -lintl -rdynamic -ldl -lnsl -lm -o program`

表 3-3 は、デバッグ・ライブラリを使用して Server-Library アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 3-3: Server-Library のデバッグ・リンクとコンパイルのコマンド

プラットフォーム	コマンド

Sun
Solaris 2.x
	`/opt/SUNWspro/bin/cc -I$SYBASE/include -L$SYBASE/devlib ¥`
	`program.c -lsrv [-lsybdb | -lct] -lcs -lcomn -ltcl -lintl ¥`
	`-lnsl -lm -o program`

プラットフォーム	コマンド
IBM	`xlc_r4 -I$SYBASE/include -L$SYBASE/devlib program.c -lsrv ¥ [-lsybdb| -lct] -lcs -lcomn -ltcl -lintl -lm -o program`
HP 9000(8xx)	`cc -I$SYBASE/include -L$SYBASE/devlib program.c -lsrv ¥ [-lsybdb	-lct] -lcs -lcomn -ltcl -lintl -linsck ¥ -Wl,-a,archive -lcl -lm -lBSD -o program`
SGI	`cc [-n32	-n64] -mips3 -I$SYBASE/include -L$SYBASE/devlib ¥ program.c -lsrv [-lsybdb	-lct] -lcs -ltcl -lcomn ¥ -lintl -lm -o program`
HP Tru64 UNIX	`cc -I$SYBASE/$SYBASE_OCS/include program.c ¥ -L$SYBASE/$SYBASE_OCS/devlib program.c ¥ -lsrv [-lsybdb	-lct] -lcs -ltcl ¥ -lcomn -lintl ¥ -lm -o program`
Linux	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsrv [-lsybdb|-lct] -lcs -lsybtcl -lcomn -lintl -rdynamic -ldl -lnsl -lm -o program`

表 3-4 は、共有ライブラリを使用して Server-Library アプリケーションのコンパイルとリンク (動的ドライバを使用) を行うためのコマンドを示します。

表 3-4: Server-Library の共有リンクとコンパイルのコマンド

プラットフォーム	コマンド
Sun Solaris 2.x	`/opt/SUNWspro/bin/cc -I$SYBASE/include -L$SYBASE/lib ¥ -R$SYBASE/lib program.c -lsrv [-lsybdb	-lct] -lcs -lcomn ¥ -ltcl -lintl -lnsl -ldl -lm -o program`
HP 9000(8xx)	`cc -I$SYBASE/include -L$SYBASE/lib program.c -lsrv ¥ [-lsybdb	-lct] -lcs -ltcl -lcomn -lintl -linsck ¥ -Wl -lcl -lm -lBSD -o program`
SGI	`cc [-n32	-n64] -mips3 -I$SYBASE/include -L$SYBASE/lib program.c ¥ -lsrv [-lsybdb	-lct] -lcs -ltcl -lcomn -lintl -lm -o program`

プラットフォーム	コマンド

HP Tru64 UNIX	`cc -I$SYBASE/$SYBASE_OCS/include program.c ¥`
	`-L$SYBASE/$SYBASE_OCS/devlib program.c ¥`
	`-lsrv [-lsybdb | -lct] -lcs -ltcl ¥`
	`-lcomn -lintl ¥`
	`-lm -o program`

注意　Open Server プログラムは、Client-Library ルーチンまたは DB-Library ルーチンを使用できます。上記の -lsrv のあとの角カッコで囲まれている情報は、DB-Library 用の -lsybdb または Client-Library 用の -lct のどちらかを選択できることを示します。

Kerberos サポート

Server-Library バージョン 11.1 以降は、ネットワークを介して通信するときに高度なセキュリティを必要とするアプリケーションに対して Kerberos セキュリティ機能をサポートします。必要な CyberSafe ソフトウェアをインストールして適切な設定作業を実行することによって、Server-Library アプリケーションはこのバージョンでサポートされる次の Kerberos セキュリティ機能を利用できます。

- ネットワーク認証
- 相互認証
- 順序不整合認証
- リプレイの検出
- 機密性
- 整合性

注意　このバージョンの Client-Library では、Kerberos サポートは Sun Solaris 2.x と HP/UX プラットフォームで提供されています。このほかのサポート情報については、リリース・ノートを参照してください。

Kerberos 機能を利用する Client-Library アプリケーションを作成して実行するには、表 3-5 に示す作業を行う必要があります。

表 3-5: CyberSafe Kerberos サポート用に必要な作業

作業	詳細
次の CyberSafe ソフトウェアをシステムにインストールする。 • CyberSafe Challenger バージョン 5.2.6 以降 • CyberSafe Application Development Kit バージョン 1.0.2	CyberSafe のマニュアルを参照。
CSFC5CCNAME 環境変数にクレデンシャル・キャッシュ・ディレクトリのロケーションを設定する。 CSFC5KTNAME 環境変数にキー・テーブル・ファイルのパスを設定する (デフォルト・キー・テーブル・ファイル以外を使用する場合)。	CyberSafe のマニュアルを参照。 クレデンシャル・キャッシュ・ディレクトリのデフォルト・ロケーションはプラットフォームごとに異なる。 デフォルト・キー・テーブル・ファイルは */krb5/v5srvtab*。
ct_con_props を使用してクレデンシャル (必要なセキュリティ機能) を設定する、または ct_con_props. を設定しないでデフォルト・クレデンシャルを使用する。	『Open Client Client-Library/C リファレンス・マニュアル』を参照。 セキュリティ機能がサポートされているかどうかを調べるには、ct_con_props と ct_config 内の CS_SUPPORTED アクション・タイプを使用する。
libtcl.cfg 設定ファイルのセキュリテイ・セクションを設定する。	『Open Client/Server 設定ガイド UNIX 版』を参照。
Sybase 提供の CyberSafe ドライバを、適切なフラグを使用して Client-Library アプリケーションにリンクする。	「Kerberos サポート」(41 ページ) を参照。
kadmin という CyberSafe ユーティリティを使用して、キー・テーブル・ファイルに必要なサーバ・プリンシパル用のキーを設定する。	CyberSafe のマニュアルを参照。

注意 セキュリティを強化するために、Sybase ではキー・テーブル・ファイルを “root” で作成し、ほかのすべてのユーザによるこのファイルへのアクセスを制限することをおすすめします。これは、サーバ・アプリケーションを起動できるのは “root” ユーザだけであることを意味します。

バルク・コピー・ルーチン

バルク・コピー・ルーチンを使用する場合は、*libblk* バルク・コピー・ライブラリをリンクする必要があります。

バルク・コピー・ライブラリをリンクするには、リンク行の先頭にある -lsrv の前に -lblk を追加します。

バルク・コピーの詳細については、『Open Client/Server Common Libraries リファレンス・マニュアル』を参照してください。

パフォーマンスについて

共有ライブラリとリンクすると、静的ライブラリとリンクする場合よりも実行プログラムが小さくなり、リンク時間も少なくて済みます。ただし、共有ライブラリとリンクされた実行プログラムは、静的ライブラリを使用してリンクされた実行プログラムよりも起動に時間がかかります。さらに、静的ライブラリとは違って共有ライブラリは実行時に使用可能でなければなりません。

最高のパフォーマンスを提供するライブラリのタイプは、それぞれのサイトの稼働条件によって決まります。

ヘッダ・ファイル

ospublic.h ヘッダ・ファイルは、すべての Open Server アプリケーションのソース・ファイルにインクルードする必要があります。ほかの必要なヘッダ・ファイルは、*ospublic.h* 内にネストされています。Bulk-Library を使用する場合は、*ospublic.h* のほかに *bkpublic.h* もインクルードしてください。

ヘッダ・ファイルの詳細については、『Open Server Server-Library/C リファレンス・マニュアル』を参照してください。

Server-Library のサンプル・プログラム

この項では、Server-Library にオンラインで提供されているサンプル・プログラムについて説明します。

オンライン・サンプル・プログラムは、C プログラムでの Server-Library ルーチンの一般的な使い方を示します。このサンプル・プログラムはサーバであり、interfaces ファイル内またはネットワーク・ディレクトリ・サービス内にそのマシンとネットワーク・アドレスを示すエントリが必要です。ディレクトリ・サービスおよび interfaces ファイルの設定方法については、『Open Client/Server 設定ガイド UNIX 版』を参照してください。

サンプル・プログラムの目的

サンプル・プログラムは、Open Server に固有の機能の例を示します。これらのプログラムは、ctosdemo 以外は、Open Server のトレーニング用ではなく、アプリケーション・プログラマのためのガイドとして設計されています。サンプル・プログラムを使用する前に、各ソース・ファイルの先頭にある説明を読んで、ソース・コードの内容を確認してください。

これらの簡単なプログラムは、実際の運用環境で使用するために作成されているものではありません。実際の運用環境で使用できるレベルのプログラムでは、エラーや特殊なケースを処理するためのコードを追加する必要があります。

これらのサンプル・プログラムで使用できるトレース・フラグについては、それぞれのサンプル・プログラムで確認してください。サンプル・プログラムを実行する方法の詳細については、*README* ファイルを参照してください。

注意 Sun Solaris では、makefile 内の *secsrv_krb* に対応するリンク行を SYBPLATFORM 値 “dce_sun_svr4” 用に更新し、CyberSafe ライブラリとのリンクを除外する必要があります。このリンク行は *csfgss_pPtr* シンボルを定義しません。このシンボルは、169 行目の GETKRBLIBS 変数で次のように定義されます。

```
KRBLIBS=" -Bdynamic -lsocket " ;; ¥
```

ロケーション

サンプル・プログラムは *$SYBASE/sample/srvlibrary* ディレクトリにあります。このディレクトリには次のファイルが含まれています。

- サンプル・プログラムのオンライン・ソース・コード
- サンプル・プログラムを構築するための *makefile*。Server-Library アプリケーションの作成を開始するときに使用します。
- サンプル・ヘッダ・ファイル *ossample.h*

- srv_connect イベント・ハンドラ
- エラー・ハンドラ
- サンプル・プログラムの構築、実行、テストの方法について説明している *README* ファイル

注意 クライアント・プログラムは、stop_srv レジスタード・プロシージャを実行することによってシングルスレッドおよびマルチスレッドのサンプル・プログラムを停止させることができます。

サンプル・プログラムの概要

次のようなサンプル・プログラムがソフトウェアに含まれています。

注意 マルチスレッドのサンプル・プログラムの場合、クライアントは stop_serv レジスタード・プロシージャを使用してサンプル・プログラムを停止します。

---

ctosdemo.c

ctosdemo.c プログラムは、Server-Library 呼び出しと Client-Library 呼び出しを使用する Open Server ゲートウェイ・アプリケーションです。クライアントからコマンドを受け取って、リモート Adaptive Server に渡し、次にリモート・サーバから結果を取り出して、クライアントに渡します。*ctosdemo.c* プログラムは、次のようなさまざまなクライアント・コマンドを処理します。

- バルク・コピー・コマンド
- カーソル・コマンド
- 動的 SQL コマンド
- 言語コマンド
- オプション・コマンド
- RPC (リモート・プロシージャ・コール)

さらに、このサンプル・プログラムは srv_attention イベント・ハンドラを呼び出すことによってクライアントからのアテンション要求に応答します。このプログラムは、各タイプのクライアント・コマンドを処理するためにイベント・ハンドラ・ルーチンを持っています。

ゲートウェイの詳細については、『Open Server Server-Library/C リファレンス・マニュアル』を参照してください。

exfds.c

exfds.c プログラムは、Open Server の全プロセスをブロックすることなく Open Server アプリケーションが外部ファイル記述子にサービスを提供できる方法を示します。このプログラムは次の処理を実行します。

- srv_capability ルーチンを使用して、現在のプラットフォームが srv_poll をサポートしているかどうかを調べます。
- 2 つの UNIX パイプをオープンします。
- srv_spawn ルーチンを使用して、srv_poll と srv_stop という 2 つのサービス・スレッドを生成します。

2 つのサービス・スレッドは、UNIX パイプにメッセージを書き込むことによって簡単なコマンドとその応答プロトコルを実装します。srv_poll は、メッセージを待っている間に Open Server がサービス・スレッドを再スケジュールできるようにするために使用されます。進行状況をモニタする情報が *srv.log* に書き込まれます。Open Server は、ソース・コード内で指定された回数だけコマンドとその応答プロトコルを実行してから SRV_STOP イベントをキューに入れます。

このサンプル・プログラムはクライアント・アプリケーションを必要としません。プログラムが正しく起動したかどうか調べるには、*srv.log* ファイルのメッセージを確認してください。

fullpass.c

fullpass.c プログラムは、Sybase の TDS (Tabular Data Stream™) passthrough モードを使用する Open Server ゲートウェイ・アプリケーションです。TDS passthrough モードの詳細については、『Open Server Server-Library/C リファレンス・マニュアル』の第 2 章にある「パススルー・モード」の項を参照してください。

イベント・ハンドラ・ルーチンは srv_recvpassthru を使用してクライアント要求を受け取り、ct_sendpassthru ルーチンを使用してこの情報を Adaptive Server に転送します。クライアント・コマンド全体がリモート・サーバに転送されると、イベント・ハンドラは ct_recvpassthru を使用してリモート・サーバから結果を読み込み、srv_sendpassthru を使用してクライアントに渡します。

このアプリケーションは、SRV_CONNECT イベント・ハンドラも含んでいます。このハンドラは、srv_getloginfo と ct_setloginfo を使用して、クライアント接続情報をリモート・サーバに転送します。次に ct_getloginfo と srv_setloginfo を使用して、接続確認情報をクライアントに渡します。TDS passthrough モードを使用するすべての Open Server アプリケーションは、その SRV_CONNECT イベント・ハンドラ内にこれらの呼び出しを含んでいる必要があります。

intlchar.c

このサンプル・プログラムは、Open Server による各国言語と文字セットの処理の例を示します。このプログラムは、Open Server アプリケーションの各国言語と文字セットのための値を初期設定してから、クライアントの要求に応じてこれらの値を変更します。

クライアントの要求は、オプション・コマンドと言語コマンドのフォーマットで渡されます。*intlchar.c* は、SRV_OPTION および SRV_LANGUAGE イベント・ハンドラに加えて SRV_CONNECT ハンドラもインストールします。

lang.c

lang.c プログラムは、srv_language イベント・ハンドラの使い方を示します。このイベント・ハンドラは、情報メッセージを使用してクライアントの言語コマンドに応答します。このとき、イベント・ハンドラは srv_sendinfo ルーチンを使用して情報メッセージをクライアントに送信します。このプログラムには、srv_connect イベント・ハンドラとエラー・ハンドラも含まれています。

言語コマンドを処理する方法の詳細については、『Open Server Server-Library/C リファレンス・マニュアル』の「言語呼び出し」の項を参照してください。

multthrd.c

multthrd.c プログラムは、次のようないくつかの Open Server マルチスレッド・プログラミング機能の例を示します。

- srv_spawn によるサービス・スレッドの作成
- クライアント接続スレッドとサービス・スレッド間でのメッセージ・キューによるスレッド間通信 (srv_getmsgq と srv_putmsgq を使用)
- スリープ・メカニズムとウェイクアップ・メカニズム (srv_sleep と srv_wakeup を使用)
- スケジューリング情報をレポートするためのコールバック・ルーチンの使用 (srv_callback を使用)

サービス・スレッドは、この Open Server アプリケーションが受け取るすべての言語クエリのログを取ります。

アプリケーションの言語ハンドラでは、クライアント・スレッドはクライアントからクエリを読み込み、メッセージ・データとしてクエリを使用してメッセージを「ロガー」というサービス・スレッドに送信します。送信後、クライアント・スレッドは待機します (srv_sleep)。サービス・スレッドは、メッセージを受け取るとクライアント・スレッドをウェイクアップします (srv_wakeup)。ロガーは連続的にループを繰り返してメッセージを待ちます。メッセージを受け取ると、クエリの内容をファイルに出力して、送信側をウェイクアップします。

ロガーとクライアント・スレッドは、スケジューリング情報を表示するために、SRV_C_RESUME、SRV_C_SUSPEND、SRV_C_TIMESLICE、SRV_C_EXIT の各コールバック・ハンドラをインストールします。*multthrd.c* プログラムは、SRV_START ハンドラ、SRV_LANGUAGE ハンドラ、SRV_CONNECT ハンドラ、コールバック・ハンドラをインストールします。

osintro.c

osintro.c プログラムは、Open Server アプリケーションの基本的なコンポーネントの例を示します。イベント・ハンドラはインストールされていません。

regproc.c

regproc.c プログラムは、Open Server バージョン 11.1 以降でのレジスタード・プロシージャの使用例を示します。アプリケーションは、起動時にいくつかのプロシージャを登録して、クライアント・コマンドを待ちます。Open Server イベント・ハンドラはインストールされません。

クライアントは RPC コマンドを送信して、*regproc.c* に定義されているレジスタード・プロシージャを実行します。

regproc.c とともに使用するためのクライアント・プログラムが、次のようにいくつか提供されています。

- *version.c* － Open Server のバージョンを返すレジスタード・プロシージャ (rp_version) を実行します。
- *dbwait.c* － DB-Library とともに実装され、レジスタード・プロシージャ rp_version が実行されるときに Open Server が通知を受けるように登録します。
- *ctwait.c* － Client-Library とともに実装され、レジスタード・プロシージャ rp_version が実行されるときに Open Server が通知を受けるように登録します。

secsrv.c

secsrv.c プログラムは、Open Server のネットワーク・ベースのセキュリティ・サービスの使用例を示します。このサンプル・プログラム内の接続ハンドラは、クライアント・スレッドのセキュリティ・プロパティを取り出し、そのセッションでどのセキュリティ・サービスがアクティブになっているかを示すメッセージをクライアントに送信します。

セキュリティ・サービスの詳細については、『Open Client/Server 設定ガイド UNIX 版』を参照してください。

sigalarm.c

sigalarm.c プログラムは、Open Server アプリケーションが UNIX の SIGALARM シグナルを使用して周期的なイベントをスケジュールする方法を示します。*sigalarm.c* は具体的には次のような処理を行います。

- srv_spawn を使用して、アラームによってウェイクアップされるまでスリープするサービス・スレッドを生成します。サービス・スレッドは、ウェイクアップされるたびに srv_log ルーチンを使用して、Open Server ログ・ファイルにメッセージを書き込みます。
- srv_signal ルーチンを使用して、SIGALARM ハンドラが呼び出されるたびにスリープ中のサービス・スレッドをウェイクアップする SIGALARM ハンドラをインストールします。*sigalarm.c* は、UNIX の alarm を呼び出すことによって、特定の間隔で SIGALARM に配信されるように要求します。

このサンプル・プログラムはクライアント・アプリケーションを必要としません。プログラムが正しく起動したかどうか調べるには、*srv.log* ファイルのメッセージを確認してください。

第 4 章

Open Client Embedded SQL/COBOL

Embedded SQL は Transact-SQL のスーパーセットであり、COBOL 言語などで作成されるアプリケーション・プログラムに Transact-SQL 文を埋め込むことができます。Embedded SQL には、すべての Transact-SQL 文に加えて、アプリケーション・プログラムで Transact-SQL を使用するために必要な拡張機能が含まれています。

Embedded SQL/COBOL は、Adaptive Server データベースに格納されているデータの取得、挿入、変更を行うための簡単な方法を提供します。

一般的な条件

サンプル・プログラムをはじめとする Embedded SQL/COBOL アプリケーションを実行するには、以下の準備が必要です。

- pubs2 サンプル・データベースがインストールされている Adaptive Server にアクセスできるようにします。pubs2 データベースをインストールする方法については、『ASE インストール・ガイド』を参照してください。
- 次の環境変数を設定します。詳細については、「付録 B 環境変数」を参照してください。
  - SYBASE
  - COBDIR
  - PATH
  - SYBPLATFORM
  - プラットフォーム固有のライブラリ・パス変数

Embedded SQL/COBOL 実行プログラムの構築

この項では、ライブラリ、リンク方法、ヘッダ・ファイルについて説明します。

ライブラリ

次の表は、すべての Embedded SQL/COBOL 機能を十分に活用するために組み込む必要のあるライブラリを示します。表の 1 行目には、すべてのプラットフォームで使用できるライブラリを示します。2 行目以降は各プラットフォームに固有なライブラリを示します。

表 4-1: Embedded SQL/COBOL 用のプラットフォーム固有ライブラリ

プラットフォーム	サポートされているライブラリ
すべてのプラットフォーム	*libcobct* − Client-Library と CS-Library に対する COBOL インタフェース (Sybase) *libct* − Client-Library (Sybase) *libcs* − CS-Library (Sybase) *libcomn* − 内部共有ユーティリティ・ライブラリ (Sybase 内部使用) *libintl* − 国際化サポート・ライブラリ (Sybase 内部使用) *libtcl* − トランスポート制御層 (Sybase 内部使用)

Embedded SQL/COBOL アプリケーションから実行プログラムを構築するには、次の 3 つの基本手順を実行します。

1 アプリケーションをプリコンパイルします。

2 プリコンパイラによって生成された COBOL ソース・コードをコンパイルおよびリンクします。

3 プリコンパイラによって生成されたストアド・プロシージャをロードします。

次の項ではこれらの手順について説明します。

アプリケーションのプリコンパイル

Embedded SQL/COBOL ソース・プログラムをプリコンパイルするための構文は、次のとおりです。

```
cobpre [-a] [-b] [-c] [-d] [-e] [-f]
    [-l] [-m] [-q] [-r] [-v] [-w] [-x] [-y]
   [-Ccompiler]
   [-Ddatabase_name]
   [-Ffips_level]
   [-G[isql_file_name]]
   [-Iinclude_directory]...
   [-Jlocale_for_charset]
   [-Ksyntax_level]
   [-L[listing_file_name]]
   [-Ninterfaces_file_name]
   [-Otarget_file_name]
```

```
[-P[password]]
[-Sserver_name]
[-Ttag_id]
[-Uuser_id]
[-Vversion_number]
[-Zlocale_for_messages]
[@file_name]
program[.ext] [program[.ext]]...
```

program は、Embedded SQL/COBOL ソース・ファイルの名前です。*program* のデフォルトの拡張子は ".pco" です。cobpre を実行すると、".cbl" 拡張子の付いた出力ファイルが生成されます。

オプションの一部には、ストアド・プロシージャの生成などの、プリコンパイラの機能を有効にするためのスイッチもあります。これらの機能はデフォルトでは「オフ」になっています。「オン」にするには、cobpre 文の行にそのオプションを指定します。このほかの文修飾子は、パスワードなど、プリプロセッサに対する値を指定します。値はオプションのあとに入力します (間にスペースを入れても入れなくてもかまいません)。

正しくないオプションを指定した場合は、プリコンパイラは使用可能なオプションのリストを表示します。

cobpre の詳細については、「付録 A コマンドおよびユーティリティ」を参照してください。

アプリケーションのコンパイルとリンク

以下の表は、UNIX オペレーティング・システムが稼働し、Sybase がサポートするプラットフォーム上で、Embedded SQL/COBOL アプリケーションのコンパイルとリンクを行うコマンドの一般的なフォーマットを示します。

表 4-2 は、非デバッグ・ライブラリを使用して Embedded SQL/COBOL アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 4-2: Embedded SQL/COBOL の非デバッグ・リンクとコンパイルのコマンド

プラットフォーム	コマンド
Sun Solaris 2.x	`cob -x program.cbl -L $SYBASE/lib -lcobct -lct -lcs -ltcl ¥` `-lcomn -lintl -ltli -lnsl -lm -o program`
HP 9000(8xx)	`cob -x program.cbl -L $SYBASE/lib -lcobct -lct -lcs -ltcl ¥` `-lcomn -lintl -linsck -lBSD -lm -o program`
IBM RS/6000	`cob -x program.cbl -L $SYBASE/lib -lcobct -lct -lcs -ltcl ¥` `-lcomn -lintl -linsck -lm -o program`
SGI	`cob -x program.cbl -L $SYBASE/lib -lcobct -lct -lcs -ltcl ¥` `-lcomn`

プラットフォーム	コマンド
HP Tru64 UNIX	cobol -ansi -names upper -x program.cbl -L $SYBASE/lib¥ -lcobct -lct -lcs -lcomn -ltcl -lintl -lm -o program

表 4-3 は、デバッグ・ライブラリを使用して Embedded SQL/COBOL アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 4-3: Embedded SQL/COBOL のデバッグ・リンクとコンパイルのコマンド

プラットフォーム	コマンド
Sun Solaris 2.x	cob -g -x program.cbl -L $SYBASE/devlib -lcobct -lct -lcs ¥ -lcomn -ltcl -lintl -ltli -lnsl -lm -o program
HP 9000(8xx)	cob -g -x program.cbl -L $SYBASE/devlib -lcobct -lct -lcs ¥ -ltcl -lcomn -lintl -linsck -lm -o program
IBM RS/6000	cob -g -x program.cbl -L $SYBASE/devlib -lcobct -lct -lcs ¥ -ltcl -lcomn -lintl -linsck -lm -o program
HP Tru64 UNIX	cobol -ansi -names upper -x program.cbl -L$SYBASE/devlib ¥ -lcobct -lct -lcs -lcomn -ltcl -lintl -lm -o program

HP Tru64 UNIX 用の新しいコンパイルとリンクのコマンド行

次の汎用コマンドを使用して、DEC COBOL 用に Embedded SQL/COBOL アプリケーションをコンパイル、リンクします。

非スレッド非リエントラント・ライブラリ

- 最適化ライブラリ ― 非スレッド、非リエントラント

```
cobol -ansi -names upper -x program.cbl ¥
 $SYBASE/$SYBASE_OCS/lib/libcobct.a ¥
 $SYBASE/$SYBASE_OCS/lib/libct.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcs.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcomn.a ¥
 $SYBASE/$SYBASE_OCS/lib/libtcl.a ¥
 $SYBASE/$SYBASE_OCS/lib/libintl.a ¥
 -lm -o program
```

- デバッグ・ライブラリ ― 非スレッド、非リエントラント

```
cobol -ansi -names upper -x program.cbl ¥
 $SYBASE/$SYBASE_OCS/devlib/libcobct.a ¥
 $SYBASE/$SYBASE_OCS/devlib/libct.a ¥
 $SYBASE/$SYBASE_OCS/devlib/libcs.a ¥
 $SYBASE/$SYBASE_OCS/devlib/libcomn.a ¥
 $SYBASE/$SYBASE_OCS/devlib/libtcl.a ¥
```

```
$SYBASE/$SYBASE_OCS/devlib/libintl.a ¥
-lm -o program
```

次の汎用コマンドを使用して、MicroFocus COBOL 用に Embedded SQL/COBOL アプリケーションをコンパイル、リンクします。

- 最適化ライブラリ ― 非スレッド、非リエントラント

```
cob -x program.cbl ¥
 $SYBASE/$SYBASE_OCS/lib/libcobct.a ¥
 $SYBASE/$SYBASE_OCS/lib/libct.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcs.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcomn.a ¥
 $SYBASE/$SYBASE_OCS/lib/libtcl.a ¥
 $SYBASE/$SYBASE_OCS/lib/libintl.a ¥
 -lm -o program
```

スレッド・リエントラント・ライブラリ

- 最適化ライブラリ ― スレッド、リエントラント

```
cobol -ansi -names upper -x program.cbl ¥
 $SYBASE/$SYBASE_OCS/lib/libcobct_r.a ¥
 $SYBASE/$SYBASE_OCS/lib/libct_r.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcs_r.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcomn_r.a ¥
 $SYBASE/$SYBASE_OCS/lib/libtcl_r.a ¥
 $SYBASE/$SYBASE_OCS/lib/libintl_r.a ¥
 -threads -lm -o program
```

- デバッグ・ライブラリ ― スレッド、リエントラント

```
cobol -ansi -names upper -x program.cbl ¥
 $SYBASE/$SYBASE_OCS/devlib/libcobct_r.a ¥
 $SYBASE/$SYBASE_OCS/devlib/libct_r.a ¥
 $SYBASE/$SYBASE_OCS/devlib/libcs_r.a ¥
 $SYBASE/$SYBASE_OCS/devlib/libcomn_r.a ¥
 $SYBASE/$SYBASE_OCS/devlib/libtcl_r.a ¥
 $SYBASE/$SYBASE_OCS/devlib/libintl_r.a ¥
-threads -lm -o program
```

ストアド・プロシージャのロード

プリコンパイラの -G フラグを使用してストアド・プロシージャを生成する場合は、プログラムを実行する前に isql 文を使用してストアド・プロシージャを Adaptive Server にロードする必要があります。生成されたスクリプトを実行するための isql 文のフォーマットは、次のとおりです。

```
isql -Uuserid -Ppassword < program.sql
```

-U フラグには Adaptive Server にログインするためのユーザ ID を指定し、-P フラグにはパスワードを指定します。

isql については、「付録A コマンドおよびユーティリティ」を参照してください。

Embedded SQL/COBOL サンプル・プログラム

Embedded SQL/COBOL プリコンパイラには、以下の項で説明するような、一般的な Embedded SQL アプリケーションの例を示す 2 つのオンライン・サンプル・プログラムが提供されています。

注意 *$SYBASE/sample/esqlcob* の内容を、もとのファイルの整合性に影響を与えないでサンプル・プログラムを自由に使用できるような「作業」ディレクトリにコピーしてから、コンパイルして実行してください。

サンプル・プログラムの目的

サンプル・プログラムは、Embedded SQL/COBOL に固有の機能の例を示しています。これらのプログラムは、Embedded SQL/COBOL のトレーニング用ではなく、アプリケーション・プログラマのためのガイドとして設計されています。サンプル・プログラムを使用する前に、各ソース・ファイルの先頭にある説明を読んで、ソース・コードの内容を確認してください。

サンプル・プログラムは編集する必要があります。プログラムをプリコンパイルする前に、ユーザ名とパスワードを Adaptive Server で有効な値に置き換えてください。変更箇所についてはプログラム内のコメントを参照してください。サンプル・プログラムを実行する方法の詳細については *README* ファイルを参照してください。

これらの簡単なプログラムは、実際の運用環境で使用するために作成されているものではありません。実際の運用環境で使用できるレベルのプログラムでは、エラーや特殊なケースを処理するためのコードを追加する必要があります。

ロケーション

サンプル・プログラムは次のディレクトリに格納されています。

```
$SYBASE/sample/esqlcob
```

このディレクトリには次のファイルが含まれています。

- サンプル・プログラムのオンライン・ソース・コード
- サンプル・プログラムを構築するための *makefile*。*makefile* は、Embedded SQL/COBOL アプリケーションの作成を開始するときに使用します。
- サンプル・プログラムの構築、実行、テストの方法について説明している *README* ファイル

注意 サンプル・プログラムの結果を表示するには、[Return] キーを押す必要があります。

例 1：データベース・クエリのためのカーソルの使い方

例 1 は、対話型クエリ・プログラムでのカーソルの使い方を示します。次にサンプル・プログラムの処理の流れを示します。このプログラムは次のように動作します。

- 本のタイプのリストを表示します。ユーザはタイプを 1 つ選択します。
- 選択されたタイプの本のすべてのタイトルを表示して、タイトル ID を要求します。
- 選択されたタイトルについての詳細情報を表示し、さらにタイトル ID を要求します。
- プロンプト画面で [Return] キーが押されると、プログラムは終了します。

例 2：テーブルのローの表示と編集

例 2 は、カーソルを使用してローを更新する方法を示します。このプログラムは次のように動作します。

- authors テーブル内のカラムをローごとに表示します。
- ユーザは au_id カラムを除くすべてのカラム内の作家情報を更新できます。ユーザがカラム情報に対して [Return] キーを押した場合は、そのカラムのデータは変更されないでもとのままになります。
- ユーザが更新を確認したあと、データを Adaptive Server に送ります。

第 5 章

Open Client Embedded SQL/C

Embedded SQL は、C などの言語で作成されたアプリケーション・プログラム内に Transact-SQL 文を埋め込むための Transact-SQL のスーパーセットです。Embedded SQL には、すべての Transact-SQL 文に加えて、アプリケーション・プログラムで Transact-SQL を使用するために必要な拡張機能が含まれています。

Embedded SQL は、Adaptive Server データベースに保管されているデータの検索、挿入、修正を行うための簡単な方法を提供します。

この章の内容は、次のとおりです。

一般的な条件

サンプル・プログラムをはじめとする Embedded SQL/C アプリケーションを実行するには、以下の準備が必要です。

- 次の環境変数を設定します。詳細については、「付録 B 環境変数」を参照してください。
  - SYBASE
  - SYBPLATFORM
  - プラットフォーム固有のライブラリ・パス変数
- pubs2 サンプル・データベースがインストールされている Adaptive Server にアクセスできるようにします。pubs2 データベースをインストールする方法については、『ASE インストール・ガイド』を参照してください。
- ファイルの所有者に対し、*sybopts.sh* の実行パーミッションを次のように設定します。

  ```
  chmod u+x sybopts.sh
  ```

- 検索パスに現在のディレクトリを追加します (まだ指定していない場合)。

  ```
  setenv PATH .:$PATH
  ```

Embedded SQL/C 実行プログラムの構築

Embedded SQL アプリケーションから実行プログラムを構築するには、次の手順に従います。

1 アプリケーションをプリコンパイルします。

2 プリコンパイラによって生成された C ソース・コードをコンパイルして、必要なファイルやライブラリとアプリケーションをリンクします。

3 プリコンパイラによって生成されたストアド・プロシージャをロードします。

次の項ではこれらの手順について説明します。

アプリケーションのプリコンパイル

ソース・プログラムをプリコンパイルするための構文は次のとおりです。

```
cpre [-a] [-b] [-c] [-d] [-e] [-f]
     [-l] [-m] [-p] [-r] [-s] [-v] [-w] [-x] [-y]
    [-Ccompiler]
    [-Ddatabase_name]
    [-Ffips_level]
    [-G[isql_file_name]]-H
    [-Iinclude_directory]...
    [-Jlocale_for_charset]
    [-Ksyntax_level]
    [-L[listing_file_name]]
    [-Ninterfaces_file_name]
    [-Otarget_file_name]
    [-P[password]]
    [-Sserver_name]
    [-Ttag_id]
    [-Uuser_id]
    [-Vversion_number]
    [-Zlocale_for_messages]
    [@file_name]
 program[.ext] [program[.ext]]...
```

program は、Embedded SQL/C ソース・ファイルの名前です。*program* のデフォルトの拡張子は ".cp" です。cpre を実行すると、".c" という拡張子の付いた出力ファイルが生成されます。

オプションの一部には、プリコンパイラの機能を有効にするためのスイッチもあります。たとえば、あるオプションはストアド・プロシージャを生成します。これらの機能はデフォルトでは「オフ」になっています。「オン」にするには、cpre 文の行にそのオプションを指定します。このほかの文修飾子は、パスワードなど、プリプロセッサに対する値を指定します。値はオプションのあとに入力します (間にスペースを入れても入れなくてもかまいません)。

正しくないオプションを指定した場合は、プリコンパイラは使用可能なオプションのリストを表示します。

プリコンパイラ・オプションの詳細については、「付録 A コマンドおよびユーティリティ」を参照してください。

アプリケーションのコンパイルとリンク

この項では、ライブラリ、リンクの方法、ヘッダ・ファイルについて説明します。

注意 Client-Library と Server-Library は、Net-Library、ディレクトリ・ドライバ、セキュリティ・ドライバの動的ロードをサポートするようになりました。この変更は、Client-Library、Server-Library、Embedded SQL アプリケーションのリンクの方法に影響します。

このバージョン以降は、アプリケーションに次のオブジェクト・ファイルを明示的にリンクする必要はありません。

- Sybase Net-Library ドライバ (HP-UX、HP Tru64 UNIX、SGI、IBM RS/6000 の場合のリンカ・オプションは -linsck、Sun Solaris 2.x の場合は -ltli)
- Sybase ディレクトリ・ドライバ (リンカ・オプションは -lddce) またはセキュリティ・ドライバ (リンカ・オプションは -lsdce)

以下の表は、UNIX オペレーティング・システムが稼働し、Sybase がサポートするプラットフォーム上で、Embedded SQL/C アプリケーションのコンパイルとリンクを行うコマンドの一般的なフォーマットを示します。

表 5-1 は、静的ライブラリを使用して Embedded SQL/C アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 5-1: Embedded SQL/C の静的リンクとコンパイルのコマンド

プラットフォーム	コマンド

Sun Solaris 2.x	`/opt/SUNWspro/bin/cc -I$SYBASE/include -L$SYBASE/lib ¥`
	`  *APP_FILES -lct* -lcs -ltcl -lcomn -lintl -Bdynamic`
	`  -lnsl -ldl -lm -o program`

IBM RS/6000	`xlc_r4 -I$SYBASE/include -L$SYBASE/lib`
	`*APP_FILES -lct* ¥`
	`  -lcs -ltcl -lcomn -lintl -lm -o program`

HP 9000(8xx)	`cc -I$SYBASE/include -L$SYBASE/lib *APP_FILES* [-W1,-a,archive] ¥`
	`  *-lct* -lcs -ltcl -lcomn -lintl -Wl,-a,default -lcl -lm ¥`
	`  -lBSD -ldld -Wl, -E, +s -o program`

SGI	`cc -o [-n32 | -n64] -mips3 -I$SYBASE/include -Bstatic *APP_FILES* ¥`
	`  -L$SYBASE/lib *-lct* -lcs -ltcl -lcomn -lintl ¥`
	`  -Bdynamic -lm -o program`

HP Tru64 UNIX	`cc -I$SYBASE/include -L$SYBASE/lib *APP_FILES -lct* -lcs -ltcl ¥` `  -lcomn -lintl -lm -o program`

プラットフォーム	コマンド
Linux	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib APP_FILES -lct -lcs -lsybtcl -lcomn -lintl -rdynamic -ldl -lnsl -lm -o program`

表 5-2 は、デバッグ・ライブラリを使用して Embedded SQL/C アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 5-2: Embedded SQL/C のデバッグ・リンクとコンパイルのコマンド

プラットフォーム	コマンド
Sun Solaris 2.x	`/opt/SUNWspro/bin/cc -I$SYBASE/include -L$SYBASE/devlib ¥ -g APP_FILES -lct -lcs -ltcl -lcomn -lintl -Bdynamic -lnsl -ldl -lm -o program`
IBM RS/6000	`xlc_r4 -I$SYBASE/include -L$SYBASE/devlib -g APP_FILES ¥ -lct -lcs -ltcl -lcomn -lintl -lm -o program`
HP 9000(8xx)	`cc -I$SYBASE/include -L$SYBASE/devlib -g APP_FILES ¥ [-Wl,-a,archive] -lct -lcs -ltcl -lcomn -lintl ¥ -Wl,-a,default -lcl -lm -lBSD -ldld -Wl, -E, +s -o program`
SGI	`cc -g [-n32 | -n64] -mips3 -I$SYBASE/include -L$SYBASE/devlib ¥ APP_FILES -lct -lcs -ltcl -lcomn -lintl -lm -o program`
HP Tru64 UNIX	`cc -I$SYBASE/include -L$SYBASE/devlib APP_FILES -lct -lcs ¥ -ltcl _oldstyle_liblookup -lcomn -lintl -lnsl -lm -o program`
Linux	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib APP_FILES -lct -lcs -lsybtcl -lcomn -lintl -rdynamic -ldl -lnsl -lm -o program`

表 5-3 は、共有ライブラリを使用して Embedded SQL/C アプリケーションのコンパイルとリンク (動的ドライバを使用) を行うためのコマンドを示します。

表 5-3: Embedded SQL/C の共有リンクとコンパイルのコマンド

プラットフォーム	コマンド
Sun Solaris 2.x	`cc -I$SYBASE/include -L$SYBASE/lib APP_FILES ¥ $SYBASE/include/sybesql.c -lct -lcs -ltcl ¥ -lcomn -lintl -ltli -lnsl -ldl -lm -o program`

プラットフォーム	コマンド

HP 9000(8xx)	`cc -I$SYBASE/include -L$SYBASE/lib` *`APP_FILES`* `¥`
	`$SYBASE/include/sybesql.c -lct -lcs -ltcl ¥`
	`-lcomn -lintl -linsck -Wl -lcl -lm -lBSD -o program`

SGI	`cc [-n32 | -n64] -mips3 -I$SYBASE/include -L$SYBASE/lib` *`APP_FILES`* `¥` *`-lct`* `-lcs -ltcl -lcomn -lintl -ldl -lm -o program`

HP Tru64 UNIX	`cc -I$SYBASE/include -L$SYBASE/lib` *`APP_FILES`* `¥`
	`-oldstyle_liblookup` *`-lct`* `-lcs -ltcl-lcomn ¥`
	`-lintl -lm -o program`

Linux	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib` *`APP_FILES`* *`-lct`* `-lcs -lsybtcl -lcomn -lintl -rdynamic -ldl -lnsl -lm -o program`

注意　*sybesql.c* ファイルをコンパイルすることによって生成されるオブジェクトは、Embedded SQL/C アプリケーションで使用されるユーティリティ・ルーチンを含んでいます。アプリケーションが正しく動作するためには、すべてのアプリケーションに *sybesql.o* をリンクしなければなりません。

- Embedded SQL/C アプリケーションのリンク行は、Client-Library アプリケーションの場合に使用するリンク行と同一です。APP_FILES はそのリンク行に、次に示す情報をその順序どおりに含む必要があります。
  - 生成されたすべての C ファイル (または、生成された C ファイルからコンパイルされた *.o* ファイル)
  - *$SYBASE/include/sybesql.o* (アップグレードしている場合は、このファイルの最新バージョンを使用していることを確認する)
- *-lct* は、コードが呼び出す Open Client と Open Server のライブラリをリンクするためのリンカ・オプションを表します。これらのオプションは、次に示すリンカ・オプションの一部または全部を次に示す順序で指定できます。

非 DCE アプリケーションの場合	DCE アプリケーションの場合
-lsrv (Server-Library ルーチン用)	-lsrv_r (Server-Library ルーチン用)
-lblk (Bulk-Library ルーチン用)	-lblk_r (Bulk-Library ルーチン用)
-lct (Client-Library ルーチン用)	-lct_r (Client-Library ルーチン用)

HP-UX システムのユーザの場合

- -W1,-a,archive オプションを使用すると、リンカは Sybase ライブラリを静的にリンクします。このオプションを指定していない場合は、Sybase ライブラリの共有バージョンが使用されます。この場合は、実行時に SH_LIB_PATH 環境変数に *$SYBASE/lib* を設定しておく必要があります。また、アプリケーション・ユーザは *$SYBASE/lib* 内のライブラリに対する read と execute のパーミッションが必要です。
- HP-UX は、アプリケーションが +s リンカ・オプションを使用してリンクされている場合を除いて、実行時に SH_LIB_PATH 環境変数を使用しません。システムが実行時に Sybase ライブラリを使用できるようにするには、+s リンカ・オプションを使用してください。-E は、実行時にドライバ・ライブラリがロードされるときに、未定義シンボル・エラーにならないようにするために必要です。詳細については、HP-UX の ld の man ページを参照してください。

HP Tru64 UNIX 用の新しいコンパイルとリンク行

次の汎用コマンドを使用して、Embedded SQL/C アプリケーションをコンパイル、リンクします。

非スレッド、非リエントラント・ライブラリ

- 最適化ライブラリ ― 非スレッド、非リエントラント

```
cc -I$SYBASE/$SYBASE_OCS/include program.c ¥
 $SYBASE/$SYBASE_OCS/lib/libct.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcs.a ¥
 $SYBASE/$SYBASE_OCS/lib/libtcl.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcomn.a ¥
 $SYBASE/$SYBASE_OCS/lib/libintl.a ¥
 -lm -o program
```

- デバッグ・ライブラリ ― 非スレッド、非リエントラント

```
cc -I$SYBASE/$SYBASE_OCS/include
-L$SYBASE/$SYBASE_OCS/devlib program.c ¥
 -lct -lcs -ltcl -oldstyle_liblookup ¥
 -lcomn -lintl -lnsl ¥
 -lm -o program
```

- 共有ライブラリ ― 非スレッド、非リエントラント

```
cc -I$SYBASE/$SYBASE_OCS/include
-L$SYBASE/$SYBASE_OCS/lib program.c ¥
 -oldstyle_liblookup -lct -lcs ¥
 -ltcl -lcomn -lintl ¥
 -lm -o program
```

スレッド・リエントラント・ライブラリ

- 最適化ライブラリ ― スレッド、リエントラント

```
cc -I$SYBASE/$SYBASE_OCS/include program.c -threads ¥
 $SYBASE/$SYBASE_OCS/lib/libct_r.a ¥
 $SYBASE/$SYBASE_OCS/lib/libcs_r.a ¥
```

```
$SYBASE/$SYBASE_OCS/lib/libtcl_r.a ¥
$SYBASE/$SYBASE_OCS/lib/libcomn_r.a ¥
$SYBASE/$SYBASE_OCS/lib/libintl_r.a ¥
-lm -o program
```

- デバッグ・ライブラリ ― スレッド、リエントラント

```
cc -I$SYBASE/$SYBASE_OCS/include
-L$SYBASE/$SYBASE_OCS/devlib -g ¥
 -threads program.c ¥
 -lct_r -lcs_r -ltcl_r -lcomn_r-lintl_r ¥
 -oldstyle_liblookup ¥
 -lm -o program
```

- 共有ライブラリ ― スレッド、リエントラント

```
cc -I$SYBASE/$SYBASE_OCS/include
-L$SYBASE/$SYBASE_OCS/lib program.c ¥
 -threads -oldstyle_liblookup ¥
 $SYBASE/$SYBASE_OCS/include/sybesql.c ¥
 -lct_r -lcs_r -ltcl_r ¥
 -lcomn_r
 -lintl_r -lnsl_r ¥
 -lm -o program
```

パフォーマンスについて

共有ライブラリとリンクすると、静的ライブラリとリンクする場合よりも実行プログラムが小さくなり、リンク時間も少なくて済みます。ただし、共有ライブラリとリンクされた実行プログラムは、静的ライブラリを使用してリンクされた実行プログラムよりも起動に時間がかかります。さらに、静的ライブラリとは違って共有ライブラリは実行時に使用可能でなければなりません。

最高のパフォーマンスを提供するライブラリのタイプは、それぞれのサイトの稼働条件によって決まります。

ストアド・プロシージャのロード

プリコンパイラの -G フラグを使用してストアド・プロシージャを生成する場合は、プログラムを実行する前に isql 文を使用してストアド・プロシージャを Adaptive Server にロードする必要があります。生成されたスクリプトを実行するための isql 文のフォーマットは、次のとおりです。

```
isql -Uuserid -Ppassword < program.sql
```

-U フラグには Adaptive Server にログインするためのユーザ ID を指定し、-P フラグにはパスワードを指定します。

isql については、「付録A コマンドおよびユーティリティ」を参照してください。

Embedded SQL/C サンプル・プログラム

Embedded SQL/C プリコンパイラには、一般的な Embedded SQL/C アプリケーションの例を示す 2 つのオンライン・サンプル・プログラムが提供されています。

サンプル・プログラムの目的

サンプル・プログラムは、Embedded SQL/C 固有の機能の例を示しています。これらのプログラムは、Embedded SQL/C のトレーニング用ではなく、アプリケーション・プログラマのためのガイドとして設計されています。サンプル・プログラムを使用する前に、各ソース・ファイルの先頭にある説明を読んで、ソース・コードの内容を確認してください。

これらの簡単なプログラムは、実際の運用環境で使用するために作成されているものではありません。実際の運用環境で使用できるレベルのプログラムでは、エラーを処理するためのコードを追加する必要があります。サンプル・プログラムを実行する方法の詳細については、*README* ファイルを参照してください。

ロケーション

サンプル・プログラムは次のディレクトリに格納されています。

```
$SYBASE/sample/esqlc
```

このディレクトリには次のファイルが含まれています。

- サンプル・プログラムのオンライン・ソース・コード
- サンプル・プログラムを構築するための *makefile*。*makefile* は、Embedded SQL アプリケーションの作成を開始するときに使用します。
- サンプル・ヘッダ・ファイル *sybsqlex.h*
- サンプル・プログラムの構築、実行、テストの方法について説明している *README* ファイル

$SYBASE/sample/esql の内容を、もとのファイルの整合性に影響を与えないでサンプル・プログラムを自由に使用できるような「作業」ディレクトリにコピーしてから、コンパイルして実行してください。

ヘッダ・ファイル

サンプル・プログラムをプリコンパイルする前に、次に示すようにサンプル・ヘッダ・ファイルを編集し、ユーザ名とパスワードを Adaptive Server で有効な値に置き換えておく必要があります。変更箇所についてはプログラム内のコメントを参照してください。

すべてのサンプル・プログラムは、サンプル・ヘッダ・ファイル *sybsqlex.h* を参照します。*sybsqlex.h* の内容は、次のとおりです。

```
/**********************************************
 *                                            *
 *  sybsqlex.h - header file for Embedded SQL/C   *
 *examples                                    *
 *                                            *
 **********************************************/

 #define USER            "user name"
 #define PASSWORD         "password"
```

すべてのサンプル・プログラムは次の行を含んでいます。

```
#include "sybsqlex.h"
```

sybsqlex.h 内では、USER は "user name"、PASSWORD は "password" と定義されています。サンプル・プログラムを実行する前に *sybsqlex.h* を編集して "user name" を Adaptive Server のログイン名に置き換え、“password” を Adaptive Server のパスワードに置き換えてください。

例 1：データベース・クエリのためのカーソルの使い方

example1.cp プログラムは、対話型クエリ・プログラムでのカーソルの使い方を示します。このプログラムは次のように動作します。

- 本のタイプのリストを表示します。ユーザはタイプを 1 つ選択します。
- 選択されたタイプの本のすべてのタイトルを表示して、タイトル ID を要求します。
- 選択されたタイトルについての詳細情報を表示し、さらにタイトル ID を要求します。
- プロンプト画面で [Return] キーが押されると終了します。

例 2：テーブルのローの表示と編集

example2.cp は、カーソルを使用してローを更新する方法を示します。このプログラムは次のように動作します。

- 作家テーブル内のカラムをローごとに表示します。
- ユーザは au_id カラムを除くすべてのカラム内の作家情報を更新できます。ユーザがカラム情報に対して [Return] キーを押した場合は、そのカラムのデータは変更されないでもとのままになります。
- ユーザが更新を確認したあと、データを Adaptive Server に送ります。

付 録　A

コマンドおよびユーティリティ

この付録は、次のユーティリティのリファレンス・ページです。

- bcp
  - bcp − ユーザ指定のフォーマットで、データベース・テーブルをオペレーティング・システム・ファイルに、またはオペレーティング・システム・ファイルをデータベース・テーブルにコピーします。
- cobpre と cpre
  - cobpre − COBOL ソース・プログラムをプリコンパイルして、ターゲット・ファイル、リスティング・ファイル、isql ファイルを生成します。
  - cpre − ソース・プログラムをプリコンパイルして、ターゲット・ファイル、リスティング・ファイル、isql ファイルを生成します。
- defncopy
  - defncopy − 指定されたビュー、ルール、デフォルト、トリガ、プロシージャ、またはレポートの定義を、データベースからオペレーティング・システム・ファイルにコピーしたり、またはオペレーティング・システム・ファイルからデータベースにコピーしたりします。
- isql
  - isql − Adaptive Server に対して SQL を対話的に解析します。

このバージョンでは、bcp、defncopy、isql ユーティリティによって生成されるメッセージが変更されています。特定の文字列を解析するスクリプト (たとえば、awk や grep) を使用してこれらのメッセージを処理する場合、新しいメッセージに対応するスクリプトの検索パターンを変更しなければならないことがあります。

bcp

説明

ユーザが指定したフォーマットで、データベース・テーブルをオペレーティング・システム・ファイルに、またはオペレーティング・システム・ファイルからデータベース・テーブルにコピーします。

構文

```
bcp [[database_name.]owner.]table_name
    {in | out} datafile
 [-c] [-E] [-n] [-N] [-v] [-X]
 [-a display_charset]
 [-A size]
 [-b batchsize]
 [-e errfile]
 [-f formatfile]
 [-F firstrow]
 [-I interfaces_file]
 [-J client_charset]
 [-K keytab_file]
 [-L lastrow]
 [-m maxerrors]
[-q datafile_charset]
 [-r row_terminator]
 [-R remote_server_principal]
 [-S server]
 [-t field_terminator]
 [-T text_or_image_size]
 [-U username]
 [-V [security_options]
 [-Y ]
[-z[language ]
[-Z security_mechanism]
```

パラメータ

database_name

コピーするテーブルがデフォルト・データベース内または *master* 内にある場合は、このパラメータはオプションとして使用できます。そうでない場合は、データベース名を指定しなければなりません。

owner

ユーザまたはデータベース所有者がコピーするテーブルを所有している場合は、このパラメータはオプションとして使用できます。所有者を指定しない場合は、bcp はユーザが所有している名前のテーブルを最初に検索します。次に、データベース所有者が所有しているテーブルを検索します。それ以外のユーザがテーブルを所有している場合は、所有者の名前を指定しなければなりません。指定しないと、コマンドは失敗します。

table_name

コピーするデータベース・テーブルまたはデータベース・ビューの名前です。

in | out

コピーの方向を示します。in は、ファイルからデータベース・テーブルへのコピーであることを示し、out は、データベース・テーブルからファイルへのコピーであることを示します。

datafile

オペレーティング・システム・ファイルの名前です。

-a *display_charset*

bcp を実行しているマシンの文字セットと異なる文字セットを使用する端末から、bcp を実行できます。-a と -J をともに使用すると、変換に必要な文字セット変換ファイル (*.xlt* ファイル) が指定できます。-a を使用するときに -J を省略できるのは、クライアントの文字セットがデフォルトの文字セットと同じ場合だけです。

-A *size*

この bcp セッションで使用するネットワーク・パケットのサイズを指定します。次に例を示します。

```
bcp -A 2048
```

この例では、bcp セッションのパケット・サイズを 2048 バイトに設定します。*size* には、default network packet size 設定変数と maximum network packet size 設定変数の間の値であり、512 の倍数である必要があります。

大量のバルク・コピー・オペレーションのパフォーマンスを向上させるには、デフォルトよりも大きなネットワーク・パケット・サイズを使用します。

-b *batchsize*

バッチごとにコピーされるデータのロー数です (デフォルトでは、1 つのバッチ処理ですべてのローがコピーされます)。バッチ処理は、バルク・コピー・インの場合にだけ適用されます。バルク・コピー・アウトには適用されません。

-c

char データ型をデータ・ファイル内の全カラムの記憶タイプとして使用して、コピー・オペレーションを実行します。このオプションは各フィールドの入力を要求しません。記憶タイプとして *char* データ型を使用し、プレフィクスなしで、デフォルトのフィールド・ターミネータとして ¥t (タブ)、デフォルトのロー・ターミネータとして ¥n (改行) を使用します。

-e *errfile*

bcp がファイルからデータベースに転送できなかったローを保管するエラー・ファイルの名前です。bcp プログラムからのエラー・メッセージは、使用している端末に表示されます。bcp がエラー・ファイルを作成するのは、このオプションが指定されたときだけです。

-E

テーブルの IDENTITY カラムの値を明示的に指定します。

デフォルトでは、IDENTITY カラムを持つテーブルにデータをバルク・コピーすると、bcp は IDENTITY カラムの一時的な値として 0 を各ローに割り当てます。これは、テーブルにデータをコピーしているときのみ有効です。bcp はデータ・ファイルから ID カラムの値を読み込みますが、サーバにはこの値を無視して送信しません。bcp が各ローをテーブルに挿入するときに、サーバは 1 から始まる連続した、ユニークな IDENTITY カラムの値をローに割り当てます。データをテーブルにコピーするときに -E フラグを指定すると、bcp はデータ・ファイルからデータを読み込んでサーバに送信します。サーバはこれらの値をテーブルに挿入します。挿入されるローの数が IDENTITY カラム値の最大値を超える場合、Adaptive Server はエラーを返します。

-E オプションは、ID カラムをデータ・ファイルにコピーするコピー・アウトの場合は無効です (-N オプションを使用しない場合)。

-f *formatfile*

同一テーブル内で前回の bcp 実行時の応答が保管してあるファイルのフル・パス名です。bcp に対して使用するフォーマットを入力すると、その応答をフォーマット・ファイルに保存するプロンプトが表示されます。フォーマット・ファイルを作成するかどうかはオプションです。デフォルトのファイル名は、*bcp.fmt* です。bcp プログラムは、データのコピー時にフォーマット・ファイルを参照することができます。したがって、前回と同じフォーマット応答を再度対話的に繰り返す必要がなくなります。このオプションは、コピー・インまたはコピー・アウトに対して以前に作成したフォーマット・ファイルを、今回も使用する必要があるときにだけ使用します。このオプションを指定していない場合、bcp はフォーマット情報を対話的に問い合わせてきます。

-F *firstrow*

コピーを開始する最初のローの番号を指定します (デフォルトは 1 番目のローです)。

-I *interfaces_file*

Adaptive Server に接続するときに検索する interfaces ファイルの名前とロケーションを指定します。-I を指定していない場合、bcp は SYBASE 環境変数で指定されたディレクトリにある interfaces ファイルを探します。

-J *client_charset*

クライアントで使用する文字セットを指定します。bcp は、フィルタを使用して *client_charset* と Adaptive Server 文字セット間で入力を変換します。

-J *client_charset* は、クライアントで使用する文字セットである *client_charset* とサーバの文字セット間の変換を Adaptive Server に要求します。

-J に引数を指定しないと、文字セットの変換は NULL に設定されます。この場合、変換は行われません。クライアントとサーバが同じ文字セットを使用する場合に、このパラメータを使用してください。

-J を省略すると、文字セットはプラットフォームのデフォルトに設定されます。デフォルトの文字セットは、クライアントが使用している文字セットと同じであるとはかぎりません。表 A-1 は、各プラットフォームのデフォルトの文字セットを示します。

表 A-1: 各プラットフォームのデフォルトの文字セット

プラットフォーム	デフォルトの文字セット
Sun、Digital、Pyramid、RS6000/AIX、その他	iso_1
HP	roman8

-K *keytab_file*

DCE セキュリティにだけ使用できます。-U オプションによって指定されるユーザ名のセキュリティ・キーを含んでいる DCE keytab ファイルを指定します。keytab ファイルは、DCE の dcecp ユーティリティを使用して作成できます。詳細については、DCE のマニュアルを参照してください。

-K オプションを指定していない場合、bcp ユーザは -U オプションで指定するユーザ名と同じユーザ名を使用して DCE にログインしなければなりません。

-L *lastrow*

コピーを終了する最後のローの番号を指定します (デフォルトは最後のローです)。

-m *maxerrors*

bcp がコピーをアボートするまでに許容されるエラーの最大数を指定します。bcp は、構築できない各ローを 1 つのエラーとしてカウントして、除外します。このオプションを指定していない場合、bcp はデフォルト値の 10 を使用します。

-n

ネイティブ (オペレーティング・システム) のフォーマットを使用してコピーを実行します。このオプションは各フィールドの入力を要求しません。ネイティブ・データ・フォーマットのファイルは人間が判読できるフォーマットにはなっていません。

警告！ データのリカバリ、サルベージ、または非常時の解析を実行するために bcp をネイティブ・フォーマットで使用しないでください。異なるハードウェア・プラットフォーム間、異なるオペレーティング・システム間、または異なるメジャー・リリースの Adaptive Server 間で、ネイティブ・フォーマットの bcp を使用してデータを転送しないでください。ネイティブ・フォーマットを使用して bcp を実行した場合、Adaptive Server に再ロードできないフラット・ファイルが作成され、データをリカバリできなくなることがあります。bcp を文字フォーマットで再実行できなければ、たとえば、テーブルのトランケートや削除、ハードウェアの損傷、データベースの削除などの場合に、データをリカバリできなくなります。

-N

IDENTITY カラムをスキップします。このオプションを使用するのは、データをコピー・インするときにホスト・データ・ファイルが IDENTITY カラム値のためのプレースホルダを含んでいない場合です。または、データをコピー・アウトするときに、ホスト・ファイルに IDENTITY カラム情報を含めないようにする場合に使用します。

データをコピー・インするときに、-N オプションと -E オプションの両方を使用することはできません。

-P *password*

Adaptive Server のパスワードを指定します。-V を指定すると、このオプションは無視されます。

-q *datafile_charset*

bcp を実行して、クライアントの文字セットと異なる文字セットを使用するファイル・システムとの間で、文字データを双方向にコピーできるようになります。-q と -J を一緒に使用して、変換に必要な文字セット変換ファイル (*.xlt* ファイル) を指定します。

日本語環境では、-q フラグは半角カタカナを全角カタカナに変換します。クライアントの日本語文字セット (sjis または eucjis) を指定するには、引数 "*zenkaku*" と -J フラグを一緒に使用します。*zenkaku.xlt* ファイルは、端末表示から Adaptive Server に変換するためだけに作成されたものです。Adaptive Server から端末に変換するためのファイルではありません。

注意 ascii_7 文字セットは、すべての文字セットと互換性があります。Adaptive Server またはクライアントの文字セットのどちらかが ascii_7 に設定されている場合、すべての 7 ビット ASCII 文字はクライアントとサーバ間でそのまま渡すことができます。その他の文字セットを使用している場合は、変換エラーが発生します。文字セットの変換の詳細については、『システム管理ガイド』を参照してください。

-r *row_terminator*

デフォルトのロー・ターミネータを指定します。

-R *remote_server_principal*

リモート・サーバのプリンシパル名を指定します。デフォルトでは、サーバのプリンシパル名はサーバのネットワーク名 (-S オプションまたは DSQUERY 環境変数で指定) と一致します。サーバのプリンシパル名とネットワーク名が同じでない場合は、-R オプションを使用する必要があります。

-S *server*

接続先の Adaptive Server の名前を指定します。-S を引数なしで指定すると、bcp は DSQUERY 環境変数で指定されるサーバを使用します。

-t *field_terminator*

デフォルトのフィールド・ターミネータを指定します。

-T *text_or_image_size*

Adaptive Server が送信する text または image データの最大長をバイト単位で指定します。デフォルトは 32K です。text または image フィールドが -T の値またはデフォルト値より大きい場合、bcp はオーバフロー部分を送信しません。

-U *username*

Adaptive Server ログイン名を指定します。*username* を指定していない場合、bcp は現在のユーザのオペレーティング・システム・ログイン名を使用します。

-V *security_options*

ネットワーク・ベースのユーザ認証を指定します。このオプションを使用する場合、ユーザはユーティリティを実行する前にネットワークのセキュリティ・システムにログインする必要があります。この場合、ユーザは -U オプションにネットワーク・ユーザ名を指定する必要があり、-P オプションに指定されたパスワードは無視されます。

-V の後にキー文字オプションの *security_options* 文字列を続けると、他のセキュリティ・サービスを有効にできます。指定できるキー文字は、以下のとおりです。

c − データ機密性サービスを有効にする

i − データ整合性サービスを有効にする

m − 接続の確立に相互認証を有効にする

o − データ・オリジン・スタンピング・サービスを有効にする

r − データ・リプレイの検出を有効にする

q − 順序不整合の検出を有効にする

-v

bcp プログラムの現在のバージョンと版権メッセージを表示します。

-X

サーバへの現在の接続で、アプリケーションがクライアント側のパスワード暗号化を使用してログインを開始するように指定します。bcp (クライアント) は、サーバに対してパスワードの暗号化が要求されていることを通知します。サーバは、bcp がパスワードを暗号化するために使う暗号化キーを返送し、パスワードを受け取ると、そのキーを使用してそのパスワードを確認します。

-Y

bcp IN の使用時に、サーバでの文字セット変換を無効にし、クライアント側で bcp によって文字セット変換を実行することを指定します。

注意　bcp OUT 使用時には、すべての文字セット変換はサーバで行われます。

-z *language*

サーバが bcp のプロンプトとメッセージの表示に使用する代替言語の公式名です。z フラグを指定しないと、bcp はサーバのデフォルトの言語を使用します。インストール時、またはインストールしたあとで Adaptive Server に言語を追加するには、langinstall ユーティリティまたは sp_addlanguage ストアド・プロシージャを使用します。

-Z *security_mechanism*

接続で使用するセキュリティ・メカニズムの名前を指定します。

セキュリティ・メカニズム名は *$SYBASE/install/libctl.cfg* 設定ファイルに定義されています。*security_mechanism* で名前を指定しない場合は、デフォルトのメカニズムが使用されます。セキュリティ・メカニズム名の詳細については、『Open Client/Server 設定ガイド UNIX 版』の *libtcl.cfg* ファイルの説明を参照してください。

例

1 次の例は、-c オプションによって *publishers* テーブルからデータを文字フォーマットでコピーします (すべてのフィールドには *char* を使用します)。-t *field_terminator* オプションは各フィールドをカンマで終了し、-r *row_terminator* オプションは各ラインを改行文字で終了します。bcp はパスワードだけを要求します。最後の "r" の前の最初の円記号は 2 番目の円記号をエスケープするので、円記号は 1 つだけ出力されます。

```
bcp pubs2..publishers out pub_out -c -t , -r ¥¥r
```

2 次の例は、あとでデータを Adaptive Server に再ロードするために、bcp を使用して publishers テーブルから *pub_out* という名前のファイルにデータをコピーします。[Return] キーを押すと、プロンプト画面に表示されたデフォルトが使用されます。publishers テーブルにデータをコピーするときも、同じプロンプトが表示されます。

```
bcp pubs2..publishers out pub_out
 パスワード：

 フィールド pub_id のファイル記憶タイプを入力してください [char]：
 フィールド pub_id のプレフィクス長を入力してください [0]：
 フィールド pub_id の長さを入力してください [4]：
 フィールド・ターミネータを入力してください [none]：

フィールド pub_name のファイル記憶タイプを入力してください [char]：
 フィールド pub_name のプレフィクス長を入力してください [1]：
 フィールド・ターミネータを入力してください [none]：

 フィールド city のファイル記憶タイプを入力してください [char]：
 フィールド city のプレフィクス長を入力してください [1]：
 フィールド・ターミネータを入力してください [none]：

 フィールド state のファイル記憶タイプを入力してください [char]：
 フィールド state のプレフィクス長を入力してください [1]：
 フィールド・ターミネータを入力してください [none]：
```

```
フォーマット情報をファイルに保存しますか ?[Y] y
 ホストファイル名 [bcp.fmt]：pub_form

 コピーしています ...

 3 ローをコピーしました。
 クロック・タイム (ms.)：合計 = 300   平均 = 1 (300.00 ロー / 秒)
```

3　保存されているフォーマット・ファイル *pub_form* を使用して、このデータを Adaptive Server にコピーするには、次のコマンドを使用します。

```
bcp pubs2..publishers in pub_out -f pub_form
```

4　使用できるデータ型のリストを表示するには、プロンプト画面で "?" を入力します。

```
フィールド 'pub_id' のファイル記憶タイプを入力してください
 ['char']:?
カラムの型が無効です。有効なタイプは次のとおりです。
 <cr>：DataServer カラムと同じ型です。
     c : char
     T : text
     i : int
     s : smallint
     t : tinyint
     f : float
     m : money
     b : bit
     d : datetime
     x : binary
     I : image
     D : smalldatetime
     r : real
     M : smallmoney
     n : numeric
     e : decimal
```

上記の例のように 1 文字だけを入力してください。

5　次の例は、VT200 端末上で使用される文字セットで作成されたデータ・ファイルを pubs2.publishers テーブルにコピーします。-q フラグは、そのデータ・ファイルを変換します。-z フラグは、bcp メッセージをフランス語で表示します。

```
bcp pubs2..publishers in vt200_data -J iso_1 -q vt200 -z
french
```

6　次の例は、パケット・サイズとして 4096 を使用して、Adaptive Server が 40K の *text* または *image* データを送信するように指定します。

```
bcp publishers out -T 40960 -A 4096
```

使用法

注意 Sybase の外部設定ファイルがあるときは、次のセクションを追加して bcp を有効にしてください。

[BCP]

System 11 用の bcp は、Client-Library を使用して構築されています。

bcp のユーザ・インタフェースは、次の点以外は変更されていません。

- 接続でのネットワーク・ベースのセキュリティ・サービスを使用可能にする、次のようなコマンド・ライン・オプションが新しく追加されました。

  -K *keytab_file*
  -R *remote_server_principal*
  -V *security_options*
  -Z *security_mechanism*

- -y *sybase_directory* オプションは無視されます。
- エラー・メッセージのフォーマットが、以前のバージョンの bcp とは異なります。以前のメッセージの値に基づいてルーチンを実行するスクリプトがある場合は、それらの書き換えが必要な場合があります。

  たとえば、転送されたローの数を示す表示メッセージが変更されています。セッションの間、このバージョンの bcp は転送されたローの合計を定期的にレポートします。このメッセージは、以前のバージョンの bcp で出力されていたメッセージ “1000 rows transferred” に置き換わるものです。

注意 以前のバージョンの bcp を使用するには、*ocs.cfg* ファイルの [bcp] セクションの CS_BEHAVIOR プロパティを、次のように設定してください。

[bcp]

CS_BEHAVIOR = CS_BEHAVIOR_100

CS_BEHAVIOR が CS_BEHAVIOR_100 に設定されていない場合、bcp 11.1 以降の機能を使用することができます。

- bcp は、データベース・テーブルとオペレーティング・システム・ファイル間でデータを転送するための便利で高速な方法を提供します。bcp は、さまざまなフォーマットのファイルの読み込みと書き込みを行うことができます。ファイルからコピー・インする場合、bcp はデータを既存のデータベース・テーブルに挿入します。ファイルにコピー・アウトする場合は、bcp はファイルの以前の内容を上書きします。
- 処理を完了すると、bcp は正常にコピーしたデータのロー数、コピーに要した合計時間、1 つのローをコピーするのに要した平均時間 (ミリ秒単位)、1 秒当たりにコピーしたロー数を表示します。

- bcp ユーティリティは、対応するターゲット・テーブルのカラムの文字長を超えるエントリを含んでいるローは挿入しません。たとえば、bcp は、300 バイトのフィールドを持つローを、文字カラムの長さが 256 バイトのテーブルには挿入しません。この場合、bcp は次のような変換エラーを表示します。

  ```
  cs_convert:CSLIB ユーザ API レイヤ：共通ライブラリ・エラー：変換
  ／オペレーションがオーバーフローしたので、結果は切り捨てられました。
  ```

  この場合ローは無視されます。また bcp は、トランケートされたデータをテーブルに挿入しません。

  文字長の要件に違反したデータを記録するには、-e *log-file name* オプションを使用して bcp を実行します。bcp は、拒否されたデータのロー番号とカラム番号、エラー・メッセージ、データを、指定したログ・ファイル内に記録します。

インデックスまたはトリガのあるテーブルのコピー

- bcp プログラムは、データと関連のあるインデックスやトリガを持たないテーブルにデータをロードするために最適化されています。bcp は、インデックスやトリガを使用せずに、ロギングを最小限にすることで、データをテーブルに最大限のスピードでロードします。ページの割り付けはログを取られますが、ローの挿入はログを取られません。

  1 つまたは複数のインデックスやトリガを持っているテーブルにデータをコピーする場合は、bcp の低速バージョンが自動的に使用されて、ローの挿入をログに取ります。これには、create table 文の一意性制約を使用して暗黙的に作成されたインデックスも含まれます。ただし、bcp は、テーブルに対して定義されるその他の整合性制約は必要としません。

  bcp の高速バージョンはログを取らずにデータを挿入します。したがって、システム管理者やデータベース所有者は、最初にシステム・プロシージャ sp_dboption, “DB”, true を設定する必要があります。オプションが true でないときに、インデックスやトリガを持っていないテーブルにデータを挿入しようとすると、Adaptive Server はエラー・メッセージを表示します。データをファイルにコピー・アウトする場合や、インデックスやトリガを持っているテーブルにデータをコピー・インする場合は、このオプションを設定する必要はありません。

注意　bcp は、インデックスまたはトリガを持っているテーブルに対する挿入をログに取るので、ログが非常に大きくなる可能性があります。dump transaction によってログをトランケートできるのは、バルク・コピーが完了し、dump database によるデータベースのバックアップが完了したあとです。

- select into/bulkcopy オプションが「オン」のときは、トランザクション・ログをダンプすることはできません。dump transaction を発行すると、代わりに dump database を使用するように指示するエラー・メッセージが表示されます。

  警告！ select into/bulkcopy フラグをオフにする前に、データベースをダンプしてください。ログが取られていないデータがデータベースに挿入されている場合、dump database を実行する前に dump transaction を実行すると、データのリカバリができなくなります。

- ログが取られていない bcp の実行速度は、dump database が実行されている間は遅くなります。
- 表 A-2 は、コピー・インのときに bcp がどのバージョンを使用するのかを示し、select into/bulkcopy オプションに必要な設定を示します。また、トランザクション・ログが保持されるかどうか、またダンプできるかどうかも示します。

表 A-2: 高速 bcp と低速 bcp の比較

	select into/bulk copy	
bcp バージョン	**on**	**off**
高速 bcp	実行可能	bcp
(対象のテーブルにインデックスやトリガがない場合)	dump transaction の実行は不可	dump transaction の実行は不可
低速 bcp	実行可能	実行可能
(1 つまたは複数のインデックスやトリガがある場合)	dump transaction の実行は不可	dump transaction は実行可能

- デフォルトでは、新しく作成されたデータベースの select into/bulkcopy オプションは「オフ」です。デフォルトの設定を変更するには、model データベースでこのオプションを「オン」にします。

注意 インデックスまたはトリガがあるテーブルにデータをコピーする場合には、パフォーマンスが大幅に低下します。非常に大量のローをコピー・インするときは、始めに drop index (または、一意性制約として作成されたインデックスの場合は alter table) および drop trigger を使用してすべてのインデックスとトリガを削除してから、データベース・オプションを設定し、データをテーブルにコピーし、インデックスとトリガを再作成し、データベースをダンプする方が速く処理できます。ただし、クラスタード・インデックス用のインデックスとトリガの構成用として、データに必要な格納領域の他に、データに必要な格納領域の約 1.2 倍のディスク領域を割り付ける必要があります。

bcp プロンプトに対する応答

-n (ネイティブ・フォーマット) または -c (文字フォーマット) オプションを使用してデータをコピー・インまたはコピー・アウトする場合、bcp はパスワード入力のプロンプトだけを表示します。ただし、-P オプションによってパスワードが指定されている場合、プロンプトは表示されません。-n、-c、または -f *formatfile* オプションのどれも指定していない場合、bcp は、テーブル内の各フィールドについて情報を入力するよう要求します。

- 各プロンプトでは、デフォルト値は角カッコで表示されます。[Return] キーを押すと、この値を選択できます。プロンプトには、次のものがあります。
  - ファイル記憶タイプ。character データ型または Adaptive Server で有効な任意のデータ型。
  - プレフィクス長 (後続のデータの長さをバイトで示す整数)。
  - ファイル内の非 NULL フィールドのデータの記憶長。
  - フィールド・ターミネータ (任意の文字列)。
  - numeric データ型と decimal データ型の位取りと精度。

  ロー・ターミネータは、テーブルまたはファイルの最後のフィールドのフィールド・ターミネータです。
- 角カッコで囲まれたデフォルトは、当該フィールドのデータ型に適切な値を示します。ファイルへコピー・アウトする場合の空き領域の最適な使用方法は、次のとおりです。
  - デフォルトのプロンプトを使用する。
  - すべてのデータをそのテーブルのデータ型でコピーする。
  - 表示どおりのプレフィクスを使用する。
  - ターミネータを使用しない。
  - デフォルトの長さを使用する。

  次の表に、デフォルトおよび代替可能な応答を示します。

表 A-3: bcp プロンプトのデフォルトと応答

プロンプト	デフォルト設定	可能な応答
ファイル記憶タイプ	次のフィールド以外のほとんどのフィールドに対してデータベースの記憶タイプを使用する。 varchar には char varbinary には binary	人間が判読できるフォーマットのファイルの作成や読み込みには char データ型。暗黙の変換がサポートされる場合は任意の CS データ型。

プロンプト	デフォルト設定	可能な応答
プレフィクス長	0 − charデータ型 (記憶タイプではない) とすべての固定長データ型で定義されるフィールドの場合 1 − その他のほとんどのデータ型の場合 2 − char として保存される binary と varbinary の場合 4 − text と image の場合	0 − プレフィクスが不要な場合。ほかのすべての場合にはデフォルトの使用を推奨。
記憶長	char と varchar の場合は、定義されている長さを使用する。char として保存される binary と varbinary の場合は、デフォルトを使用する。その他のすべてのデータ型の場合は、トランケーションやデータのオーバフローを避けるために、必要な最大長を使用する。	デフォルト値以上の値を推奨。
フィールド・ターミネータまたはロー・ターミネータ	なし。	最大 30 文字、または次のいずれか。 ¥t タブ ¥n 改行 ¥r 復帰改行 ¥0 null ターミネータ ¥ 円記号

- bcp は、そのネイティブ (データベース) データ型、または暗黙の変換が当該データ型に対してサポートされる任意のデータ型のどちらかとして、データをファイルにコピー・アウトできます。bcp は、ユーザ定義のデータ型をそのベース・データ型または暗黙の変換がサポートされる任意のデータ型としてコピーします。データ型の変換の詳細については、『Open Client Client-Library/C リファレンス・マニュアル』の cs_convert に関する情報を参照してください。

  注意 常に同じデータ型があるとはかぎらないので、異なるバージョンの Adaptive Server からデータをネイティブ・フォーマットでコピーする場合は注意が必要です。

- プレフィクス長は、各データ値の長さをバイト単位で表現する 1 バイト、2 バイト、または 4 バイトの整数で、ホスト・ファイル内のデータ値の直前に位置します。

- データベース内で char、nchar、binary として定義されるフィールドは、データベース内で定義された全長に達するまで、常にスペース (binary の場合は null バイト) が埋め込まれます。timestamp データは binary(8) として扱われます。

  varchar と varbinary フィールド内のデータがコピー・アウトに指定する長さよりも長いと、bcp は、ファイル内のデータを指定の長さにトランケートします。

- フィールド・ターミネータ文字列は、30 文字まで指定できます。特に一般的なターミネータは、タブ (“¥t” と入力し、最後のカラム以外のすべてのカラムに使用される) と改行 (“¥n” と入力し、ローの最終フィールドに使用される) です。その他、“¥0” (null ターミネータ)、“¥” (円記号)、“¥r” (復帰改行) があります。ターミネータを選択するときは、それと同じ文字がほかの文字データ部分に存在しないものにします。たとえば、タブを含んでいる文字列に対してタブをターミネータとして使用すると、bcp は、どのタブが文字列の最後を表すのかを区別できません。bcp は常に最初に見つけたタブをターミネータと見なすので、この場合は間違ったものを見つけることになります。

  ターミネータまたはプレフィクスが存在する場合は、実際に転送されるデータの長さに影響します。ファイルにコピー・アウトされるエントリの長さが記憶長に足りないときは、その直後にターミネータが続くか、次のフィールドのためのプレフィクスが続きます。この場合、エントリに記憶長分の埋め込みは行われません (char、nchar、binary データは、Adaptive Server から返されるときすでに、いっぱいの長さまで埋め込みが行われています)。

  ファイルからコピー・インする場合、データは「長さ」プロンプトで指定されたバイト数がコピーされるか、ターミネータが検出されるまで転送されます。指定された長さのバイト数の転送が終了すると、残りのデータはターミネータが検出されるまでフラッシュされます。ターミネータがない場合、テーブルの記憶長が使用されます。

- 次の表は、ファイル内のデータに対するプレフィクス長、ターミネータ、フィールドの長さの関係を示します。“P” は格納テーブル内のプレフィクスを示します。“T” はターミネータ、ダッシュ (- -) は付加されるスペースを示します。"..." は各フィールドに対してパターンを繰り返すことを示します。各カラムのフィールド長は 8 で、“string” はそれぞれ 6 文字のフィールドを表します。

表 A-4: Adaptive Server の char データ

	プレフィクス長 = 0	プレフィクス長 1、2、または 4
ターミネータなし	string--string--.	Pstring--Pstring--.
ターミネータ	string--Tstring--T.	Pstring--TPstring--T.

表 A-5: char 記憶タイプに変換されるその他のデータ型

	プレフィクス長 = 0	プレフィクス長 1、2、または 4
ターミネータなし	string--string--.	PstringPstring...
ターミネータ	stringTstringT...	PstringTPstringT...

- ファイル内のカラムの記憶タイプと長さは、データベース・テーブル内のカラムの記憶タイプと長さと同じである必要はありません (コピー・インされるタイプとフォーマットがデータベース・テーブルの構造と互換性がないと、コピーは失敗します)。
- ファイル記憶長は、通常はターミネータやプレフィクスを除く、カラムに転送されるデータの最大サイズを示します。
- テーブルにデータをコピーする場合は、bcp はカラムに対して定義されているデフォルトとユーザ定義のデータ型を調べます。ただし、bcp はできるだけ高速にデータをロードするためにルールを無視します。
- bcp は、null 値を含むデータ・カラムを可変長と認識するので、プレフィクス長またはターミネータのどちらかを使用して各ローのデータの長さを記述してください。
- ネイティブ・フォーマットでホスト・ファイルに書き込まれたデータは、その精度をすべて保持します。datetime 値および float 値は、文字フォーマットに変換される際も精度を保持します。Adaptive Server は、*money* 値を通貨単位の 1 万分の 1 の精度まで保持します。しかし、money 値が文字フォーマットに変換される場合は、文字フォーマット値は、近似値 2 桁しか記録されません。
- 文字フォーマットのデータをファイルからデータベース・テーブルにコピーする前に、『ASE リファレンス・マニュアル』のデータ型に関する項で説明されているデータ型の入力規則をチェックしてください。bcp を使用してデータベースにコピーされる文字データは、これらの規則に従わなければなりません。特に、*(yy)yymmdd* のような区切り文字のない日付形式の場合は、年が最初に指定されていないと、オーバフロー・エラーになる可能性があります。
- 使用している端末と異なる端末を使用するサイトにホスト・データ・ファイルを送るには、送り先に、そのファイルを作成するために使用した *datafile_charset* を通知する必要があります。

メッセージ

変換テーブルのロード中にエラーが発生しました。

-a または -q パラメータで指定した文字変換ファイルが存在しないか、入力した名前が正しくありません。

cobpre と cpre

説明

cpre は C ソース・プログラムをプリコンパイルして、ターゲット・ファイル、リスティング・ファイル、isql ファイルを生成します。

cobpre は COBOL ソース・プログラムをプリコンパイルして、ターゲット・ファイル、リスティング・ファイル、isql ファイルを生成します。

構文

cobpre|cpre [-a] [-b] [-c] [-d]
[-e] [-f] [-l] [-m] [-p (cpre)] [-q (cobbre)]
[-r][-s (cpre)] [-t (cobpre)] [-v] [-w] [-x] [-y]
[-Ccompiler]
[-D*database*_name]
[-F*fips_level*]
[-G[*isql_file_name*]]
[-H[*server_name*]]
[-I*include_directory*]...
[-J*locale_for_charset*]
[-K*syntax_level*]
[-L[*listing_file_name*]]
[-N*interfaces_file_name*]
[-O*target_file_name*]
[-P[*password*]]
[-S*server_name*]
[-T*tag_id*]
[-Vversion_*number*]
-Z*locale_for_messages*]
[@*file_name*]
program[.*ext*] [*program*[.*ext*]]... Parameters

パラメータ

-a

トランザクション間で、カーソルをオープンしたままにできるようにします。カーソルとトランザクションの詳細については、『ASE リファレンス・マニュアル』を参照してください。このオプションを使用していない場合、カーソルは set close on endtran が有効であるかのように動作します。これは ANSI 準拠の動作です。

-b

fetch 文で一般的に使用されるホスト変数アドレスの再バインドを無効にします。このオプションを使用していない場合は、Embedded SQL/C プログラム内で再バインドを行わないように指定しないかぎり、fetch 文があるたびに再バインドが行われます。

-b オプションは、Embedded SQL プリコンパイラの 11.1 バージョンと 10.0.x バージョンで次のような相違点があります。

- バージョン 11.1 の cpre では、norebind 属性が適用されるのはカーソルのすべての fetch 文です。ただし、カーソルの宣言が -b オプションによってプリコンパイルされていた場合です。
- 10.0.x バージョンの cpre では、norebind 属性が適用されるのは、-b オプションによってプリコンパイルされた各 Embedded SQL ソース・ファイル内にあるすべての fetch 文です。この場合、カーソルがこのオプションによって宣言されたかどうかは関係ありません。

-c

Client-Library のデバッグ機能をオンにします。-c オプションを使用すると、Client-Library は ct_debug に対する呼び出しを生成します。このオプションは、アプリケーションの開発のときには役立ちますが、最終的にアプリケーションを配布するときにはオフにしてください。

-d

区切り識別子を使用しないことを示します。アプリケーションは、文字データを二重引用符 (“ “) で囲んでサーバに送信できます。

-e

exec sql connect 文を処理するときに、外部設定ファイルを使用して接続の設定を行うように、Client-Library に指示します。このオプションの指定がなければ、プリコンパイラは Client-Library の関数呼び出しを生成して接続の設定を行います。外部設定ファイルと CS_CONFIG_BY_SERVERNAME プロパティについては、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。

-f

ESQL ソース・ファイル内に含まれている SQL 文が、ANSI SQL-89 規格に準拠しているかどうかを検査します。SQL Server 名と有効なユーザ名およびパスワードが使用された場合、プリコンパイラはサーバに接続して、標準に準拠しているかどうか検査するために、すべての静的 SQL 文をサーバに渡します。-f は、アプリケーションが稼働しているときに、SQL 文が規格に準拠しているかどうかをサーバが検査し続けるようにするためのコードを生成します。

-l

#line ディレクティブを生成しないようにします。

-m

アプリケーションを Sybase のオートコミット・モードで実行します。オートコミット・モードとは、トランザクションが連鎖されていないことを意味します。明示的な開始トランザクションと終了トランザクションが必要であり、なければそれぞれの文は即座にコミットされます。このオプションを指定しないと、アプリケーションは ANSI 形式の連鎖トランザクション・モードで実行されます。

-p

入力ホスト変数を持つモジュール内の SQL 文ごとに別々のコマンド・ハンドルを生成し、各コマンド・ハンドルに対して継続バインドを有効にします。このオプションを使用すると、入力パラメータがあって繰り返し実行されるコマンドについて、格納領域の使用量が増加して、各コマンドの初回の応答時間が長くかかるのと引き替えに、パフォーマンスを改善できます。

ホスト文字列変数が空のときに NULL 文字列の代わりに空の文字列が挿入されないと動作しないアプリケーションは、-p オプションがオンになっていると正しく機能しません。継続バインドを実装しているので、Embedded SQL は Client-Library プロトコル (NULL 文字列を挿入する) を回避することができません。

-q
一重引用符 (') ではなく、二重引用符 (") を使用してコードを生成します。

-r
繰り返し読み出しを無効にします。このオプションを使用しないと、set transaction isolation level 3 文が生成されて、connect 文の最中に実行されます。

-s
プリコンパイルされたファイル内に静的関数宣言をインクルードします。

-t
入力ファイルのフォーマットを端末モードに指定します (HP Tru64 UNIX の場合)。

-v
プリコンパイラのバージョン情報だけを表示します。プリコンパイルは実行しません。

-w
プリコンパイラの警告メッセージと情報メッセージを画面やリスティング・ファイルに出力しないようにします。

-x
Client-Library が外部設定ファイルを使用するようにします。『Open Client Client-Library リファレンス・マニュアル』にある CS_EXTERNAL_CONFIG プロパティ、および『Open Client Embedded SQL/COBOL プログラマーズ・ガイド』にある INITIALIZE_APPLICATION 文の説明を参照してください。

-y
CS_TEXT データ型と CS_IMAGE データ型を入力ホスト変数として使用できるようにします。実行時に、データはサーバに送信される文字列に直接挿入されます。サポートされるのは静的 SQL 文だけです。動的 SQL に対する入力パラメータとして、text および image を使用することはできません。コマンド文字列でのこの引数の置換は、-y コマンド・ライン・オプションが使用されたときだけ実行されます。

-C*compiler*
対象のホスト言語コンパイラを指定します。

cpre － 指定できる値は、ANSI 規格の C では “ansi_c”、Kernighan and Ritchie C では “kr_c” です。このオプションを指定しないと、*ansi_c* がデフォルトのターゲット・ホスト言語コンパイラとして使用されます。

cobpre － *mf_byte* (バイト整列データを使用する Micro Focus COBOL コンパイラの場合)。プリコンパイラにこの引数を指定するときは、Micro Focus COBOL コンパイラに -C NOIBMCOMP 引数を指定してください。

mf_word (ワード整列データを使用する Micro Focus COBOL コンパイラの場合)。プリコンパイラにこの引数を指定するときは、Micro Focus COBOL コンパイラに -C IBMCOMP 引数を指定してください。このオプションを指定しないと、*mf_byte* がデフォルト・コンパイラとして使用されます。

-D*database_name*

解析対象のデータベースの名前を指定します。このオプションは、プリコンパイル時に SQL セマンティックを検査する場合に使用します。-G を指定すると、use *database* コマンドが *filename.sql* ファイルの先頭に追加されます。このオプションを指定しない場合、プリコンパイラは Adaptive Server のデフォルト・データベースを使用します。

-F*fips_level*

Embedded SQL ソース・ファイル内に含まれている SQL 文が、ANSI SQL-89 または SQL-92E 規格に準拠しているかどうかを検査します。プリコンパイラがサーバに接続できるように、Adaptive Server 名と有効なユーザ名およびパスワードを指定してください。-F オプションは、アプリケーションが稼働しているときに、SQL 文が規格に準拠しているかどうかをサーバが検査し続けるようにするためのコードを生成します。指定できる値は、SQL89 または SQL92E です。

-G[*isql_file_name*] (オプション)

該当する SQL 文にストアド・プロシージャを生成して、それらを isql ファイルに保管します。ストアド・プロシージャをロードするには、この isql ファイルについて isql を実行してください。複数の入力ファイルがある場合は、-G を使用することはできますが、引数を指定することはできません。

複数の入力ファイルがある場合、または引数を指定しない場合、デフォルトのターゲット・ファイル名 (1 つまたは複数) は、拡張子 ".sql" を付加した (または入力ファイルの拡張子に置き換えた) 入力ファイル名になります。また、ストアド・プロシージャのタグ ID を指定するときは、-T*tag_id* オプションも参照してください。-G オプションを使用しない場合、ストアド・プロシージャは生成されません。

-H[*server_name*] (オプション)

HA フェールオーバ・オプションを呼び出します。プライマリ・サーバで障害が発生した場合、セカンダリ・サーバが使用されます。コマンド・ライン引数パーサが -H フラグを判読し、フェールオーバ・サーバ名を格納し、必要な CT-Lib 文を生成して HA フェールオーバを実装します。このオプションを使用するには、セカンダリ・サーバが実行中であり、HA フェールオーバ・サーバとして interfaces ファイルに正しく指定されていることが必要です。

-I*include_directory*

Embedded SQL がインクルード・ファイルを探すディレクトリを指定します。このオプションはいくつでも指定できます。Embedded SQL は、コマンド・ラインに指定された順序に従って、複数のディレクトリを探します。このオプションを指定しないときのデフォルトは、*$SYBASE/include* ディレクトリおよび現在の作業ディレクトリです。

-J*locale_for_charset*

プリコンパイルするソース・ファイルの文字セットを指定します。このオプションの値は、ロケール・ファイル内のエントリに対応するロケール名でなければなりません。-J を指定しない場合、プリコンパイラはソース・ファイルがプリコンパイラのデフォルト文字セットを使用していると解釈します。

デフォルトとして使用する文字セットを決定するために、プリコンパイラはロケール名を調べます。プリコンパイラは、次の環境変数を次の順序に従って検索します。

- LC_ALL
- LANG

LC_ALL が定義されている場合、プリコンパイラはこの値をロケール名として使用します。LC_ALL が定義されていなくて LANG が定義されている場合、プリコンパイラはその値をロケール名として使用します。

LC_ALL と LANG のどちらも定義されていない場合、プリコンパイラは「デフォルト」のロケール名を使用します。

プリコンパイラは *locales* ファイル内からロケール名を探して、そのロケール名に対応する文字セットをデフォルトの文字セットとして使用します。

-K*syntax_level*

実行する構文検査のレベルを指定します。次のどちらかが選択できます。

- NONE

  検査を行いません (プリコンパイルのときに Adaptive Server を必要としません)。

- SYNTAX

  プリコンパイルのときに Adaptive Server によって構文が検査されますが、Embedded SQL 文によって参照されるデータベース・オブジェクトが存在している必要はありません。

- SEMANTIC

  Embedded SQL 文の構文とセマンティックが正しいかどうか検査されます。SQL 文によって参照されるデータベース・オブジェクトが存在している必要があります。セマンティックの検査は、コマンド・ライン・オプションに関係なく、型が CS_TEXT または CS_IMAGE の入力ホスト変数を含む文については実行されません。

-L[*listing_file_name*] (引数はオプション)
1 つまたは複数のリスティング・ファイルを生成します。リスティング・ファイルは、行番号が付けられた各行のあとに、エラーがある場合は該当するエラー・メッセージが続く入力ファイル・フォーマットのファイルです。複数の入力ファイルがある場合は、-L を指定することはできますが、引数を指定することはできません。

複数の入力ファイルがある場合、または引数を指定しない場合、デフォルトのリスティング・ファイル名 (1 つまたは複数) は、".lis" 拡張子を付加した (または、入力ファイルの拡張子を置き換えた) 入力ファイル名になります。このオプションを指定しない場合は、リスティング・ファイルは生成されません。

-N*interfaces_file_name*
プリコンパイラに対して interfaces ファイルの名前を指定します。このオプションを指定しない場合は、デフォルトの interfaces ファイルとして *$SYBASE/interfaces* が使用されます。

-O*target_file_name*
ターゲット・ファイルとなる出力ファイルの名前を指定します。複数の入力ファイルがあるときは、このオプションは使用できません (デフォルトのターゲット・ファイル名が割り当てられます)。このオプションを指定しないと、デフォルトのターゲット・ファイル名は、入力ファイル名に次のような拡張子が付いたファイル名になります。

cpre － 拡張子 *.c* を付加 (または、入力ファイル名の拡張子を置き換える)

cobpre － 拡張子 *.cbl* を付加 (または、入力ファイル名の拡張子を置き換える)

-P[*password*] (-U*user_id* オプションとともに使用。引数はオプション)
プリコンパイル時に SQL 構文を検査するための Adaptive Server パスワードを指定します。引数を付けずに -P を指定するか、引数としてキーワード NULL を使用すると、null ("") パスワードが指定されます。-P を使用しないで -U*user_id* オプションを使用すると、プリコンパイラはパスワードの入力を要求します。

-S*server_name*
プリコンパイル時に SQL 構文を検査する Adaptive Server の名前を指定します。このオプションを使用しない場合は、DSQUERY 環境変数の値がデフォルト Adaptive Server 名として使用されます。

-T*tag_id* (-G オプションとともに使用)
生成されるストアド・プロシージャ・グループ名の最後に付加するタグ ID (最大 3 文字) を指定します。

たとえば、コマンドの一部として -Tdbg を入力すると、生成されるストアド・プロシージャは、入力ファイルの名前にタグ ID として *dbg* が付加された名前 (*program_dbg;1*、*program_dbg;2* など) になります。

プログラマはタグ ID を使用することによって、使用中の可能性がある既存の生成されたストアド・プロシージャに影響を与えずに、既存のアプリケーションに対する変更をテストできます。このオプションを使用しない場合は、ストアド・プロシージャ名にタグ ID は付加されません。

-U*user_id*

Adaptive Server のユーザ ID を指定します。

このオプションを使用すると、プリコンパイル時に SQL 構文を検査できます。このオプションは、SQL 文を解析のためだけにサーバに渡すようにプリコンパイラに指示します。サーバが構文エラーを検出すると、エラーがレポートされてコードは生成されません。-P[*password*] を使用しないで -U オプションを使用すると、Embedded SQL はパスワードの入力を要求します。

このオプションを使用しない場合は、プリコンパイラはサーバに接続しないか、ターゲット・ファイルを生成するために必要とされる入力ファイルの SQL 構文検査だけを実行します。

-V*version_number*

Client-Library のバージョン番号を指定します。

cpre − バージョン番号は、CS_VERSION_100 以降でなければなりません。この値は、ct_init 関数に対するバージョン番号引数として有効な値の 1 つと一致する必要があります。ct_init 関数については、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。

cobpre − COBOL の場合、バージョン番号は *cobpub.cbl* からの値の 1 つと一致する必要があります。

このオプションを指定しないときのデフォルトは、プリコンパイラで使用できる Client-Library の最新のバージョン (Open Client/Open Server バージョン 11.1 では CS_ VERSION_110) になります。

-Z*locale_for_messages*

プリコンパイラがメッセージ用に使用する言語と文字セットを指定します。-Z の指定がないと、プリコンパイラはそのデフォルト言語と文字セットをメッセージ用に使用します。

プリコンパイラは、メッセージ用のデフォルトとして使用する言語と文字セットを次のようにして決定します。

1　ロケール名を探します。プリコンパイラは、次の環境変数を次の順序に従って検索します。

- LC_ALL
- LANG
  - LC_ALL が定義されている場合、プリコンパイラはこの値をロケール名として使用します。LC_ALL が定義されていなくて LANG が定義されている場合、プリコンパイラはその値をロケール名として使用します。
  - LC_ALL と LANG のどちらも定義されていない場合、プリコンパイラは「デフォルト」のロケール名を使用します。

2　*locales* ファイル内からロケール名を探して、そのロケール名に対応する言語と文字セットをメッセージ用のデフォルトとして使用します。

program[.ext] [program[.ext]] . . .

Embedded SQL/C ソース・プログラムの入力ファイル名です。入力ファイル名は必要な数だけいくつでも入力できます。ファイル名の形式と長さは、オペレーティング・システムの規則に違反しないかぎり、どのようなものでもかまいません。ターゲット・ファイルと複数の入力ファイルについては、「コメント」の項を参照してください。

@file_name

上記のコマンド・ライン引数のいずれかを含んでいるファイルを指定するために使用します。プリコンパイラは、すでに指定されている引数に加えてこのファイルに含まれている引数を読み込みます。*@file_name* で指定するファイル内にプリコンパイルするファイルの名前が含まれている場合は、この引数はコマンド・ラインの最後に置いてください。

例

1 プリコンパイラを実行します (ANSI 準拠)。

```
cobpre | cpre program.pco|.pc
```

2 生成されたストアド・プロシージャと FIPS フラグを使用して、プリコンパイラを実行します (ANSI 準拠)。

```
cobpre | cpre -G -f program1.pco|.pc program2.pco|.pc
```

3 トランザクション間で開いたままのカーソルを持つ 2 つの入力ファイルに対して、プリコンパイラを実行します (ANSI 非準拠)。

```
cobpre | cpre -a program1.pco|.pc program2.pco|.pc
```

4 プリコンパイラのバージョン情報だけを表示します。

```
cobpre | cpre -v
```

使用法

DCE 版のプリコンパイラ

DCE 版のプリコンパイラ実行プログラムは、DCE がインストールされているマシンで使用するように提供されています。

DCE マシン上で通常の実行プログラムを実行することはできますが、Sybase クライアント・アプリケーションのデフォルト・ディレクトリ・サービスとして DCE を設定しているときは、サーバに接続することはできません (デフォルトのディレクトリ・サービスは *$SYBASE/install/libtcl.cfg* ファイルの設定によって決まります。詳細については、『Open Client/Server 設定ガイド UNIX 版』を参照してください)。

- オプション引数

  -G[*isql_file_name*]、-L[*listing_file_name*]、-P[*password*] に引数を指定しないで使用するときは、コマンド・ライン上にほかのオプション、キーワード NULL、または二重引用符 (“”) がその次になければなりません。これらのオプションの次に、入力ファイル名を指定することはできません。指定すると、プリコンパイラはその入力ファイル名をオプション引数であると見なします。

- ANSI 規格

  cobpre コマンドと cpre コマンドのデフォルトは、ANSI 規格の動作に設定されます。

- -a、-c、-d、-f、-m、-r オプションは connect 文だけに影響します。ソース・ファイルが接続文を含んでいない場合、または -x オプションを使用する場合は、これらのオプションは効果がありません。

- ターゲット・ファイル

  cpre − デフォルトのターゲット・ファイル名は *program.c* です。入力ファイルが 1 つだけの場合は、-O *target_file_name* を使用してターゲット・ファイル名を指定できます。複数の入力ファイルがある場合は、デフォルトのターゲット・ファイル名は、*first_input_file.c*、*second_input_file.c* などになります。

  cobpre − デフォルトのターゲット・ファイル名は、入力ファイル名に拡張子 ".cbl" (Micro Focus COBOL の場合) を付加した (または、入力ファイル名の拡張子を置き換えた) ファイル名になります。入力ファイルが 1 つだけの場合は、-O *target_file_name* を使用してターゲット・ファイル名を指定できます。複数の入力ファイルがある場合、デフォルトのターゲット・ファイル名は、*first_input_file.cbl*、*second_input_file.cbl* などになります。

- オプションのフォーマット

  オプションは、引数の前にスペースがあってもなくてもかまいません。たとえば、次のどちらのフォーマットでも使用できます。

  ```
  -Tdbg
  ```

  または

  ```
  -T dbg
  ```

- 複数の入力ファイル

  プリコンパイラは複数の入力ファイルを処理できます。ただし、-O*target_file_name* オプションを使用することはできません。デフォルトのターゲット・ファイル名を使用する必要があります (上記の「ターゲット・ファイル」を参照)。-G[*isql_file_name*] オプションを使用するときは、引数を指定できません。デフォルトの *isql* ファイル名は、*first_input_file.sql*、*second_input_file.sql* などになります。-L[*listing_file_name*] オプションを使用するときも、引数を指定することはできません。デフォルトのリスティング・ファイル名は、*first_input_file.lis*、*second_input_file.lis* などになります。

アプリケーションの開発

この項では、Embedded SQL アプリケーションを開発するための、一般的に使用される手順について説明します。この手順は、稼働条件に合うように適応させることが必要な場合もあります。

1 構文検査とデバッグのために、-c、-D*database_name*、-P[*password*]、-S*server_name*、-U*user_id* オプションを使用してプリコンパイラを実行します。-G[*isql_file_name*] は使用しないでください。プログラムのコンパイルとリンクを実行して、構文が正しいかどうか確認します。

2 必要な修正をすべて行います。-D*database_name*、-P[*password*]、-S*server_name*、-U*user_id*、-G[*isql_file_name*]、-T*tag_id* オプションを使用してプリコンパイラを実行し、テスト・プログラム用のタグ ID を持つストアド・プロシージャを生成します。テスト・プログラムをコンパイルしてリンクします。次のコマンドを使用して、ストアド・プロシージャをロードします。

```
isql -P[password] -Sserver_name -Uuser_id ¥
     <isql_file_name
```

プログラム上でテストを実行します。

3 プログラムを修正し、-D*database_name*、-P[*password*]、-S*server_name*、-U*user_id*、-G[*isql_file_name*] オプションを使用してプリコンパイラを実行します (ただし、-T オプションは使用しないでください)。プログラムをコンパイルしてリンクします。次のコマンドを使用して、ストアド・プロシージャをロードします。

```
isql -P[password] -Sserver_name -Uuser_id ¥
     <isql_file_name
```

これで、最終的な配布用プログラムを実行する準備が完了しました。

cobpre と *cpre* のデフォルト

次の表は、cobpre と cpre ユーティリティのオプションとデフォルトを示します。

表 A-6: cobpre と cpre のデフォルトとオプション

オプション	オプションを使用しない場合のデフォルト
-C*compiler*	cpre − *ansi_c* コンパイラ。 cobpre − *mf_byte* コンパイラ。
-D*database_name*	Adaptive Server のデフォルト・データベース。
-G[*isql_file_name*]	ストアド・プロシージャは生成されない。
-H[*server_name*]	プライマリ・サーバで障害が発生した場合に使用されるフェールオーバ・サーバ名。
-I*include_directory*	デフォルト・ディレクトリは *$SYBASE/include*。
-J*locale_for_charset*	プラットフォームごとに異なる。
-L[*listing_file_name*]	リスティング・ファイルは生成されない。

オプション	オプションを使用しない場合のデフォルト
-O*target_file_name*	デフォルトのターゲット・ファイル名は、*.c* 拡張子 (cpre の場合) または *.cbl* 拡張子 (cobpre の場合) を付加した (または、入力ファイル名の拡張子を置き換えた) 入力ファイル名になる。
-N*interfaces_file_name*	*$SYBASE/interfaces* ファイル。
-P[*password*]	-U*user_id* を使用しないときはパスワードの入力を要求される。
-S*server_name*	デフォルトの Adaptive Server 名は DSQUERY 環境変数から取得される。
-T*tag_id*	-G を使用して生成するストアド・プロシージャ名にタグ ID は付加されない。
-U*user_id*	プリコンパイラは Adaptive Server に接続しない。または、ターゲット・ファイルを生成するために必要とされる構文検査だけを実行する。
-V*version_number*	バージョン 11.1 の場合は CS_VERSION_110。
-Z*locale_for_messages*	プラットフォームと環境によって異なる。

defncopy

説明

指定されたビュー、ルール、デフォルト、トリガ、プロシージャの定義を、データベースからオペレーティング・システム・ファイルに、またはオペレーティング・システム・ファイルからデータベースにコピーします。

注意 defncopy ユーティリティは、Report Workbench™ を使用して作成されたレポートやテーブル定義をコピーすることはできません。

構文

```
defncopy
        [-v] [-X]
        [-a display_charset]
        [-I interfaces_file]
        [-J [client_charset]]
        [-K keytab_file]
        [-P password]
        [-R remote_server_principal]
        [-S [server]]
        [-U username]
        [-V [security_options]]
        [-z language]
        [-Z security_mechanism]
        {in filename dbname |    out filename dbname
        [owner.]objectname [[owner.]objectname...] }
```

パラメータ

in | out

データベースに対する定義のコピー方向を指定します。たとえば、“in” を指定すると、定義がデータベースにコピーされます。

filename

定義コピーの送信元または送信先であるオペレーティング・システム・ファイルの名前を指定します。コピー・アウトを行うと、既存のファイルはすべて上書きされます。

dbname

定義コピーの送信元または送信先であるデータベースの名前を指定します。

objectname

defncopy がコピー・アウトするデータベース・オブジェクトの名前を指定します。定義をデータベースにコピー・インするときは、オブジェクトを指定しないでください。

-a *display_charset*

defncopy が実行されるマシンの文字セットと異なる文字セットの端末から defncopy を実行できるようにします。-a と -J を一緒に使用して、変換に必要な文字セット変換ファイル (*.xlt* ファイル) を指定します。-J なしで -a を使用するのは、クライアントの文字セットがデフォルトの文字セットと同じ場合だけです。

-I *interfaces_file*

Adaptive Server に接続するときに検索する interfaces ファイルの名前とロケーションを指定します。-I の指定がない場合、defncopy は SYBASE 環境変数によって指定されるディレクトリにある interfaces ファイルを探します。

-J *client_charset*

クライアントで使用する文字セットを指定します。フィルタによって、*client_charset* と Adaptive Server の文字セットとの間で入力が変換されます。

-J *client_charset* は、クライアントで使用する文字セットである *client_charset* とサーバの文字セット間の変換を Adaptive Server に要求します。

-J に引数を指定しないと、文字セットの変換は NULL に設定されます。この場合、変換は行われません。クライアントとサーバが同じ文字セットを使用する場合に、このパラメータを使用してください。

-J を省略すると、文字セットはプラットフォームのデフォルトに設定されます。デフォルトの文字セットは、クライアントが使用している文字セットと必ず同じであるとはかぎりません。文字セットおよび関連するフラグの詳細については、『システム管理ガイド』と『System Administration Guide Supplement』参照してください。

注意 ascii_7 文字セットは、すべての文字セットと互換性があります。Adaptive Server またはクライアントの文字セットのどちらかが ascii_7 に設定されている場合、すべての 7 ビット ASCII 文字はクライアントとサーバ間でそのまま渡すことができます。その他の文字セットを使用している場合は、変換エラーが発生します。文字セットの変換の詳細については、『システム管理ガイド』を参照してください。

-K *keytab_file*

DCE セキュリティにだけ使用できます。-U オプションによって指定されるユーザ名のセキュリティ・キーを含んでいる DCE keytab ファイルを指定します。keytab ファイルは、DCE の dcecp ユーティリティを使用して作成できます。詳細については、DCE のマニュアルを参照してください。

-K オプションを指定しない場合、defncopy のユーザは -U オプションで指定されているユーザ名と同じユーザ名を使用して DCE にログインしなければなりません。

-P *password*

パスワードを指定できるようにします。-V を指定すると、このオプションは無視されます。

-R *remote_server_principal*

リモート・サーバのプリンシパル名を指定します。デフォルトでは、サーバのプリンシパル名はサーバのネットワーク名 (-S オプションまたは DSQUERY 環境変数で指定) と一致します。サーバのプリンシパル名とネットワーク名が同じでない場合は、-R オプションを使用する必要があります。

-S *server*

接続先の Adaptive Server の名前を指定します。-S の指定がない場合、defncopy は DSQUERY 環境変数で指定されたサーバを探します。

-U *username*

ログイン名を指定できるようにします。ログイン名では、大文字と小文字を区別します。*username* を指定しない場合、defncopy は現在のユーザのオペレーティング・システム・ログイン名を使用します。

-V *security_options*

ネットワーク・ベースのユーザ認証を指定します。このオプションを使用する場合、ユーザはユーティリティを実行する前にネットワークのセキュリティ・システムにログインする必要があります。この場合、ユーザは -U オプションにネットワーク・ユーザ名を指定する必要があり、-P オプションに指定されたパスワードは無視されます。

-V の後にキー文字オプションの *security_options* 文字列を続けると、他のセキュリティ・サービスを有効にできます。指定できるキー文字は、以下のとおりです。

c − データ機密性サービスを有効にする

i − データ整合性サービスを有効にする

m − 接続の確立に相互認証を有効にする

o − データ・オリジン・スタンピング・サービスを有効にする

r − データ・リプレイの検出を有効にする

q − 順序不整合の検出を有効にする

-v

defncopy のバージョン番号と版権メッセージを表示して、オペレーティング・システムに戻ります。

-X

サーバへの現在の接続で、アプリケーションがクライアント側のパスワード暗号化を使用してログインを開始するように指定します。defncopy (クライアント) は、サーバに対してパスワードの暗号化が必要であることを通知します。サーバは暗号化キーを送り返し、defncopy はそれを使用してパスワードを暗号化します。サーバはパスワードを受け取ると、そのキーを使用してパスワードの認証を行います。

defncopy がクラッシュすると、パスワードを含むコア・ファイルが作成されます。暗号化オプションを使用していない場合、パスワードは、コア・ファイルにプレーン・テキストで表示されます。暗号化オプションを使用した場合、パスワードは表示されません。

-z *language*

サーバが defncopy のプロンプトとメッセージを表示するために使用する代替言語の公式名を指定します。-z フラグが指定されていない場合、defncopy はサーバのデフォルト言語を使用します。インストール時、またはインストールしたあとで Adaptive Server に言語を追加するには、langinstall ユーティリティまたは sp_addlanguage ストアド・プロシージャを使用します。

-Z *security_mechanism*

接続で使用するセキュリティ・メカニズムの名前を指定します。

セキュリティ・メカニズム名は *$SYBASE/install/libctl.cfg* 設定ファイルに定義されています。*security_mechanism* で名前を指定しない場合は、デフォルトのメカニズムが使用されます。セキュリティ・メカニズム名の詳細については、『Open Client/Server 設定ガイド UNIX 版』の *libtcl.cfg* ファイルの説明を参照してください。

例

```
defncopy -Usa -P -SMERCURY in new_proc stagedb
```

定義を *new_proc* ファイルから MERCURY サーバ上のデータベース stagedb へコピーします。MERCURY への接続は、ユーザ名 “sa”、パスワード NULL で確立されます。

```
defncopy -S -z french out dc.out employees sp_calccomp sp_vacation
```

“sp_calccomp” オブジェクトと “sp_vacation” オブジェクトの定義を SYBASE サーバ上の “employees” データベースから *dc.out* ファイルへコピーします。メッセージとプロンプトは「フランス語」で表示されます。ユーザは、パスワードを入力するように要求されます。

使用法

- defncopy プログラムは、オペレーティング・システムから直接起動します。defncopy を使用すると、ビュー、ルール、デフォルト、トリガ、プロシージャ用の定義 (create 文) をデータベースからオペレーティング・システム・ファイルにコピー・アウトするときに、非対話型操作でコピーを実行できます。または、指定したファイルからすべての定義をコピー・インします。

  定義をコピー・アウトするには、*sysobjects* テーブルおよび *syscomments* テーブルに対する select パーミッションが必要です。オブジェクト自体のパーミッションは必要ありません。

- コピー・インするオブジェクトのタイプについては、適切な create パーミッションを持つ必要があります。コピー・インされたオブジェクトはコピーするユーザが所有します。ユーザの代わりに定義をコピーするシステム管理者は、再構築されるデータベース・オブジェクトに対してそのユーザが正しくアクセスできるように、そのユーザとしてログインしなければなりません。

- in *filename* または out *filename*、およびデータベース名は必須であり、明確に指定される必要があります。コピー・アウトする場合は、オブジェクト名とその所有者の両方を表すファイル名を使用してください。

- defncopy は、コピー・アウトする各定義を次のようなコメントで終了します。

  ```
  /* ### DEFNCOPY: END OF DEFINITION  */
  ```

  defncopy を使用してデータベースにコピーするオペレーティング・システム・ファイル内の定義をアセンブルする場合、各定義はこの “END OF DEFINITION” という文字列を使用して終了しなければなりません。

- defncopy に対して指定する値がシェルにとって意味を持つ文字を含んでいる場合は、それらを引用符で囲んでください。

警告！ 100 文字を超える長いコメントが create 文の前にあると、defncopy が失敗することがあります。

System 11 の defncopy は、Client-Library を使用して構築されています。DCE が Sybase クライアント・アプリケーションのデフォルト・ディレクトリ・サービスである場合、またはバルク・コピー・セッションで DCE セキュリティ・サービスを使用する場合は、defncopy を使用してください。defncopy のユーザ・インタフェースは、次の点以外は変更されていません。

- 接続でのネットワーク・ベースのセキュリティ・サービスを使用可能にする、次のようなコマンド・ライン・オプションが新しく追加されました。

  -K *keytab_file*
  -R *remote_server_principal*
  -V *security_options*
  -Z *security_mechanism*

- -y *sybase_directory* オプションは無視されます。

isql

説明

Adaptive Server に対する対話型の SQL パーサです。

構文

```
isql [-b] [-e] [-F] [-n] [-p] [-v] [-X] [-Y]
 [-a display_charset]
 [-A size]
 [-c cmdend]
 [-D database]
 [-h headers]
 [-H hostname]
 [-i inputfilename]
 [-I interfaces_file]
 [-J client_charset]
 [-K keytab_file]
 [-l login_timeout]
 [-m errorlevel]
 [-o outputfilename]
 [-P password]
 [-R remote_server_principal]
 [-s colseparator]
 [-S server]
 [-t timeout]
 [-U username]
 [-V [security_options]]
 [-w columnwidth]
 [-z language]
 [-Z security_mechanism]
```

パラメータ

-a *display_charset*

isql が実行されるマシンの文字セットと異なる文字セットの端末から isql を実行できるようにします。-a と -J をともに使用すると、変換に必要な文字セット変換ファイル (*.xlt* ファイル) が指定できます。-a を使用するときに -J を省略できるのは、クライアントの文字セットがデフォルトの文字セットと同じ場合だけです。

-A *size*

この isql セッションに使用するネットワーク・パケットのサイズを指定します。次に例を示します。

```
isql -A 2048
```

この例は、この isql セッションのパケット・サイズを 2048 バイトに設定します。確認するには、次のように入力します。

```
select * from sysprocesses
```

値は *network_pktsz* という見出しの下に表示されます。

size は、default network packet size 設定変数と maximum network packet size 設定変数の間の値で、512 の倍数を指定してください。

readtext や writetext オペレーションのように I/O 集約型のオペレーションを実行する場合は、デフォルトよりも大きいパケット・サイズを使用してください。

Adaptive Server のパケット・サイズの設定や変更は、リモート・プロシージャ・コールのパケット・サイズには影響しません。

-b － テーブル・ヘッダを出力しないようにします。

-c *cmdend*

コマンド・ターミネータを再設定します。デフォルトでは、各コマンドを終了させて、“go” を 1 行に単独で入力することによって、Adaptive Server に各コマンドを送信します。コマンド・ターミネータを再設定する場合は、SQL の予約語や制御文字は使用してはいけません。? ( ) [ ] $ などのシェル・メタ文字は、必ずエスケープしてください。

-D *database*

isql のセッションを開始するデータベースを選択します。

-e

入力内容をエコーします。

-E *editor*

デフォルト・エディタ以外のエディタを指定します。

-F

FIPS フラガを使用可能にします。このオプションを使用すると、Adaptive Server は送信される標準外の SQL コマンドすべてにフラグを付けます。

-h *headers*

カラム見出しから次のカラム見出しまでの間に出力されるローの数を指定します。デフォルトでは、クエリ結果のセットごとに 1 回だけ見出しが出力されます。

-H *hostname*

クライアント・ホスト名を設定します。

-i *inputfilename*

isql への入力として使用するオペレーティング・システム・ファイルの名前を指定します。このファイルには、コマンド・ターミネータを含むようにしてください (デフォルトでは “go”)。

次のようにパラメータを指定します。

-i *inputfile*

これは次と同等です。

< *inputfile*

-i を使用して、コマンド・ラインにパスワードを指定しない場合、Adaptive Server はパスワードの入力を要求します。< *inputfile* を使用して、コマンド・ラインにパスワードを指定しない場合は、入力ファイルの最初の行にパスワードを指定してください。

-I *interfaces_file*

Adaptive Server に接続するときに検索する interfaces ファイルの名前とロケーションを指定します。-I を指定していない場合、isql は、SYBASE 環境変数によって指定されたディレクトリ下の *ini* ディレクトリ内にある interfaces ファイル (Windows プラットフォームでは *sql.ini*) を探します。

-J *client_charset*

クライアントで使用する文字セットを指定します。-J *client_charset* は、クライアントで使用する文字セットである *client_charset* とサーバの文字セット間の変換を Adaptive Server に要求します。フィルタによって、*client_charset* と Adaptive Server 文字セット間で入力を変換します。

-J に引数を指定しないと、文字セットの変換は NULL に設定されます。この場合、変換は行われません。クライアントとサーバが同じ文字セットを使用する場合に、このパラメータを使用してください。

-J の指定を省略すると、文字セットはプラットフォームのデフォルトに設定されます。デフォルトの文字セットは、クライアントが使用する文字セットと同じであるとはかぎりません。

-K *keytab_file*

DCE セキュリティにだけ使用できます。-U オプションによって指定されるユーザ名のセキュリティ・キーを含んでいる DCE keytab ファイルを指定します。keytab ファイルは、DCE の dcecp ユーティリティを使用して作成できます。詳細については、DCE のマニュアルを参照してください。

-K オプションを指定しない場合、isql のユーザは -U オプションで指定するユーザ名と同じユーザ名を使用して DCE にログインしなければなりません。

-l *login_timeout*

Adaptive Server に接続する場合の最大タイムアウト値を指定します。デフォルトは 60 秒です。この値は、サーバがログインの要求に応答するのを isql が待つ時間に対してだけ影響します。コマンド処理のタイムアウト時間を指定するには、-t *timeout* パラメータを使用します。

-m *errorlevel*

エラー・メッセージの表示をカスタマイズします。指定の重大度レベル以上のエラーの場合には、メッセージ番号、ステータス、エラー・レベルを表示し、エラー・テキストは表示しません。指定した重大度より低いレベルのエラーでは、何も表示されません。

- *curread* (現在の読み込みレベル) は、このセッション中に読み込むことができるデータの初期レベルです。*curread* は、*curwrite* よりも高いレベルでなければなりません。
- *curwrite* (現在の書き込みレベル) は、このセッション中に書き込むすべてのデータに 適用される初期 sensitivity レベルです。
- *maxread* (読み込みレベルの最大値) は、データを読み込むことができる最大レベルです。この値は、マルチレベル・ユーザとしてこのセッション中に *curread* を設定できる上限値です。*maxread* よりも高いレベルでなければなりません。

- *maxwrite* (書き込み込みレベルの最大値) は、データを書き込むことができる最大レベルです。この値は、マルチユーザとしてこのセッション中に *curwrite* に設定できる上限値です。*maxwrite* は、*minwrite* と *curwrite* よりも高いレベルでなければなりません。
- *minwrite* (書き込みレベルの最小値) は、データを書き込むことができる最小レベルです。この値は、マルチユーザとしてこのセッション中に *curwrite* に設定できる下限値です。*minwrite* は、*maxwrite* と *curwrite* よりも低いレベルでなければなりません。
- *label_value* は、システム上で使用される、人間の目で判読できるフォーマットで表現された実際のラベル値 (たとえば “Company Confidential Personnel”) です。

-n

入力行から行番号とプロンプト記号 (>) を削除します。

-o *output_filename*

isql からの出力を保管するオペレーティング・システム・ファイルの名前を指定します。

-o *outputfile*

これは次と同じです。

> *outputfile*

-p

パフォーマンスの統計値を出力します。

-P *password*

現在の Adaptive Server パスワードを指定します。-V を指定すると、このオプションは無視されます。パスワードは大文字と小文字が区別され、6 ～ 30 文字の範囲で指定できます。

-R *remote_server_principal*

リモート・サーバのプリンシパル名を指定します。デフォルトでは、サーバのプリンシパル名はサーバのネットワーク名 (-S オプションまたは DSQUERY 環境変数で指定) と一致します。サーバのプリンシパル名とネットワーク名が同じでない場合は、-R オプションを使用する必要があります。

-s *colseparator*

カラム・セパレータ文字をリセットします。デフォルト・カラム・セパレータ文字はブランクです。オペレーティング・システムに対して特別な意味を持つ文字 (| ; & <> など) を使用するには、それらを引用符で囲むか、前に円記号を付けます。

-S *server*

接続先の Adaptive Server の名前を指定します。-S の指定がないと、isql は DSQUERY 環境変数で指定されたサーバを探します。

-t *timeout*

コマンドがタイムアウトになるまでの秒数を指定します。タイムアウト値の指定がないと、コマンドは永久に実行を続けます。これは、isql 内から発行されたコマンドに影響するもので、接続時間には影響しません。

-U *username*

ログイン名を指定します。ログイン名は、大文字と小文字を区別します。

-V *security_options*

ネットワーク・ベースのユーザ認証を指定します。このオプションを使用する場合、ユーザはユーティリティを実行する前にネットワークのセキュリティ・システムにログインする必要があります。この場合、ユーザは -U オプションにネットワーク・ユーザ名を指定する必要があり、-P オプションに指定されたパスワードは無視されます。

-V の後にキー文字オプションの *security_options* 文字列を続けると、他のセキュリティ・サービスを有効にできます。指定できるキー文字は、以下のとおりです。

c − データ機密性サービスを有効にする

i − データ整合性サービスを有効にする

m − 接続の確立に相互認証を有効にする

o − データ・オリジン・スタンピング・サービスを有効にする

q − 順序不整合の検出を有効にする

r − データ・リプレイの検出を有効にする

-v

使用している isql ソフトウェアのバージョンと版権メッセージを表示します。

-w *columnwidth*

出力画面の幅を設定します。デフォルトでは、80 文字です。出力行が画面幅いっぱいになった場合は、複数の行に分割されます。

-X

サーバへのログイン接続をクライアント側のパスワード暗号化を使用して開始します。isql (クライアント) は、サーバに対してパスワードの暗号化が必要であることを通知します。サーバは、isql がパスワードを暗号化するために使う暗号化キーを返送し、パスワードを受け取ると、そのキーを使用してそのパスワードを確認します。

isql が失敗すると、システムはユーザのパスワードを含んでいるコア・ファイルを生成します。暗号化オプションを使用していない場合、パスワードは、コア・ファイルにプレーン・テキストで表示されます。暗号化オプションを使用した場合、パスワードは表示されません。

-Y

連鎖トランザクションを使用するように Adaptive Server に指示します。

-z *language*

isql のプロンプトとメッセージの表示に使用する代替言語の公式名です。-z フラグを指定しないと、isql はサーバのデフォルト言語を使用します。インストール時、またはインストールしたあとで Adaptive Server に言語を追加するには、langinstall ユーティリティまたは sp_addlanguage ストアド・プロシージャを使用します。

-Z *security_mechanism*

接続で使用するセキュリティ・メカニズムの名前を指定します。

セキュリティ・メカニズム名は Sybase インストール・ディレクトリの中の *ini* サブディレクトリにある *libtcl.cfg* 設定ファイルで定義されます。*security_mechanism* で名前を指定しない場合は、デフォルトのメカニズムが使用されます。セキュリティ・メカニズム名の詳細については、『Open Client/Server 設定ガイド UNIX 版』の *libtcl.cfg* ファイルの説明を参照してください。

例

例 1

```
isql
Password:

 1>select *
 2>from authors
 3>where city = "Oakland"
4>go
```

コマンドを実行します。

例 2

```
isql -Ujoe -Pabracadabra
1>select *
2>from authors
3>where city = "Oakland"
4>vi
```

クエリを編集できるようになります。ファイルに書き込みを行って保存すると、isql に戻ります。クエリが表示されます。クエリを実行するには go を入力します。

例 3

```
isql -U alma Password:
1>select *
2>from authors
3>where city = "Oakland"
4>reset
5>quit
```

reset によってクエリ・バッファがクリアされます。quit を入力すると、オペレーティング・システムに戻ります。

使用法

- isql を対話型で使用するには、オペレーティング・システムのプロンプト画面で isql コマンド (および任意のオプション・フラグ) を入力します。isql プログラムは、SQL コマンドを受け取り、それを Adaptive Server に送信します。結果は、フォーマットされ、標準出力に出力されます。isql を終了するには、quit または exit を使用します。
- デフォルトのコマンド・ターミネータ go で始まる行を入力するか、または -c オプションを使用する場合は、ほかのコマンド・ターミネータで始まる行を入力してコマンドを終了します。コマンド・ターミネータのあとに、コマンドを実行する回数を指定する整数を指定できます。たとえば、あるコマンドを 100 回実行する場合は、次のように入力します。

  ```
  select x = 1
   go 100
  ```

  結果は、実行の終了時に 1 回表示されます。
- コマンド・ラインにオプションを複数回入力した場合、isql は最後の値を使用します。たとえば、次のコマンドを入力したとします。

  ```
  isql -c. -csend
  ```

  -c の 2 番目の値 “send” は、最初の値 “.” を上書きします。これによって、設定したすべてのエイリアスを無効にすることができます。
- 現在のクエリ・バッファに関してエディタを呼び出すには、行の最初の単語としてエディタ名を入力します。呼び出し可能な使用エディタを定義するには、EDITOR 環境変数で指定します。EDITOR 環境変数を定義していない場合のデフォルトは vi です。

  行頭に “!!” を付けてコマンドを入力すると、オペレーティング・システム・コマンドを実行できます。EDITOR 環境変数を定義しないで、この方法を使用して代替エディタを呼び出すこともできます。
- 既存のクエリ・バッファをクリアするには、行に reset とだけ入力します。isql は、未処理の入力内容をすべて破棄します。入力行の任意の場所で [Ctrl + c] を押すことによって、現在のクエリをキャンセルして isql のプロンプト画面に戻ることもできます。
- isql によって実行されるクエリを含んでいるオペレーティング・システム・ファイルを次のようにして読み込みます。

  ```
  isql -U alma -P****** < input_file
  ```

  このファイルには、コマンド・ターミネータが含まれていなければなりません。結果は端末に表示されます。次のようにして、クエリを含むオペレーティング・システム・ファイルを読み込み、結果を別のファイルに書き込むことができます。

  ```
  isql -U alma -P****** < input_file > output_file
  ```
- isql のフラグを使用する場合には、大文字と小文字を区別してください。
- isql は float または real データを丸めて、小数点以下 6 桁までを表示します。

- isql を対話型で使用するときは、次のコマンドを使用して、オペレーティング・システム・ファイルをコマンド・バッファに読み込みます。

  ```
  :r filename
  ```

  ファイル内にコマンド・ターミネータを含めないで、編集を終わったあとにターミネータを対話的に入力してください。

- isql によって Adaptive Server に渡される Transact-SQL 文には、コメントを入れることができます。コメントは、次の例に示すように “/*” と “*/” で囲みます。

  ```
  select au_lname, au_fname
   /*retrieve authors' last and first names*/
   from authors, titles, titleauthor
   where authors.au_id = titleauthor.au_id
   and titles.title_id = titleauthor.title_id
   /*this is a three-way join that links authors
  **to the books they have written.*/
  ```

  go コマンドをコメントにする場合は、コマンドが行の先頭にならないようにします。次に例を示します。

  ```
  /*
  **go
  */
  ```

  下記のようにはしないでください。

  ```
  /*
  go
  */
  ```

isql

パフォーマンスを向上させるために、isql は、Client-Library を使用して構築されています。isql_r は isql と同じものですが、DCE が Sybase クライアント・アプリケーションのデフォルト・ディレクトリ・サービスである場合、またはバルク・コピー・セッションで DCE セキュリティ・サービスを使用する場合は、isql_r を使用してください。

isql のユーザ・インタフェースは、次の点以外は変更されていません。

- 5701 サーバ・メッセージ (「データベースが変更されました」) は、ログイン後または use database コマンドの発行後には表示されません。
- 次の 2 つのオプション・フラグが新しく追加されました。

  -b − カラム・ヘッダを出力しないようにします。
  -D *database* − isql が使用する起動時のデータベースを選択します。

- 接続でのネットワーク・ベースのセキュリティ・サービスを使用可能にする、次のようなコマンド・ライン・オプションが追加されました。

  -K *keytab_file*
  -R *remote_server_principal*
  -V *security_options*
  -Z *security_mechanism*

- エラー・メッセージのフォーマットが、以前のバージョンの isql とは異なります。これらのメッセージの内容に基づいたルーチンを実行するスクリプトは、修正が必要な場合があります。
- -y *sybase_directory* オプションは削除されました。

isql 内で使用できる追加コマンド

表 A-7: isql セッション・コマンド

コマンド	説明
reset	クエリ・バッファをクリアする
quit または exit	isql を終了する
vi	エディタを呼び出す
!! *command*	オペレーティング・システム・コマンドを実行する

参照

『ASE リファレンス・マニュアル』の sp_addlanguage、sp_addlogin、sp_configure、sp_defaultlanguage、sp_droplanguage、sp_helplanguage

付 録　B

環境変数

この付録では、Sybase アプリケーションの正しいコンパイルおよび動作のために必要な環境変数の値について説明します。設定しなければならない環境変数は、使用するアプリケーションによって異なります。また、次のような環境変数が含まれます。

- SYBASE － Sybase インストール・ディレクトリのパスを設定します。
- DSQUERY － Adaptive Server または Open Server の名前を設定します。
- DSLISTEN － Open Server の名前を設定します。
- SYBPLATFORM － 使用するプラットフォームと、DCE サービスを使用するかどうかによって適切な値を設定します。この環境変数の適切な設定値については、表 B-1 を参照してください。
- 表 B-1 にリストされているプラットフォーム固有のライブラリ・パス変数を *$SYBASE/lib* に設定して、共有 (動的) ライブラリにリンクしているプログラムを実行してください。デバッグ・モードでプログラムを実行するときは、プラットフォーム固有のライブラリ・パス変数を *$SYBASE/devlib* に設定してください。

  ESQL/COBOL アプリケーションの場合は、*$COBDIR/coblib* ディレクトリのロケーションをインクルードします。

表 B-1: SYBPLATFORM とライブラリ・パス

プラットフォーム	SYBPLATFORM の設定	プラットフォーム固有のライブラリ・パス変数
Sun Solaris 2.x	“sun_svr4”	LD_LIBRARY_PATH
Sun Solaris 2.x (ネイティブ・スレッドを使用)	“nthread_sun_svr4”	
IBM RS/6000	“rs6000”	LIBPATH
IBM RS/6000 (ネイティブ・スレッドを使用)	“nthread_rs6000”	
HP 9000(8xx)	“hpux”	SH_LIB_PATH
HP 9000(8xx) (ネイティブ・スレッドを使用)	“nthread_hpux”	
SGI － 32 ビット版	“sgi32”	LD_LIBRARY_PATH
SGI － 64 ビット版	“sgi64”	
SGI － 32 ビット版 (ネイティブ・スレッドを使用)	“dce_sgi32” “nthread_sgi64”	
SGI － 64 ビット版 (ネイティブ・スレッドを使用)		

プラットフォーム	SYBPLATFORM の設定	プラットフォーム固有のライブラリ・パス変数
HP Tru64 UNIX (ネイティブ・スレッドを使用)	“axposf” “nthread_axposf”	LD_LIBRARY_PATH
Linux (ネイティブ・スレッドを使用)	“nthread_linux”	LD_LIBRARY_PATH

Embedded SQL/COBOL アプリケーションの場合は、上記の環境変数のほかに、以下の環境変数を設定する必要があります。

- COBDIR － COBOL コンパイラのパスを設定します。
- PATH － *$COBDIR/bin* を追加します。

索引

B

C

D

E

F

G

H

I

L

M

O

P

R

S

T

U

お

か

こ

さ

す

せ

そ

た

に

は

ひ

ふ

へ

も

ゆ

ら