
# Net Express

# UNIX オプション ユーザー ガイド

Micro Focus NetExpress™

# UNIX オプションユーザーガイド

Micro Focus®

第 2 版
1998年 10月

# 序文

本書は、NetExpress UNIX オプションについて説明します。UNIX オプションにより、NetExpress 上で作成されたアプリケーションを UNIX システムに展開することができます。

## 対象読者

本書は、UNIX システムに移植する COBOL プログラムを作成するために NetExpress を使用するすべてのプログラマと設計者を対象としています。ビジネスコンピューティング、Microsoft Windows の使用法、UNIX システムの使用と管理に関する知識があることを前提としています。

## 関連マニュアル

- NetExpress および COBOL システムの他の構成要素のオンラインヘルプ
- Micro Focus COBOL for UNIX バージョン 3.2 と 4.0、Object COBOL for UNIX V4.1 と Server Express に付属のマニュアルセット

## 表記規則

- Enter は、キャリッジリターンまたは Enter キーを示します。コマンドを入力する箇所では、Enter キーは明示的には示されていません。行の終わりで Enter キーを押すことが暗黙的に示されています。
- 16 進数は引用符で囲まれ、数字の前に小文字の x または h が付きます。たとえば、x"9D"、 h"03FF"です。x は 16 進数が文字列を表すときに使用されます。h は数値を表すときに使用されます。
- COMP-X および COMP-5 データ型では PIC 99 ではなく PIC X が使用されます。PIC 99 と異なり、PIC X はデータ項目の長さを示すため、COMP-X の使用法をより明確に示します。COMP-X は、指定したバイト数のバイナリ項目を定義します。
- キートップとメニューの選択肢は、太字で示されています。
- 使用する環境によっては、本書の表示と画面の表示が多少異なる場合があります（たとえば、バージョン番号など）。この違いは、ソフトウェアの動作には影響しません。
- 本書で使用するキーは、すべての環境で使用できるとは限りません。ステータスキーやファンクションキーなどのキーの使用を示している場合には、物理的なキーストロークではなく、そのキーを論理的に押し下げ、離すことを意味します。指定されたキーがご使用の環境でサポートされていない場合には、*リリースノート*を参照して同等のキーを見つけてください。
- 用語「ウィンドウ」は、画面上で区切られた領域を示します。通常は画面全体よりも小さい領域です。用語「Windows」は、Microsoft Windows95 またはそれ以降のバージョン、および Windows NT を示します。

- オンラインヘルプは印刷されたマニュアルではありません。メニュー [ヘルプ] を選択するか、またはダイアログボックスで [ヘルプ] ボタンを押して、必要なヘルプ情報を表示してください。

  用語「COBOL for UNIX」は、Micro Focus COBOL for UNIX システムのすべてのサポートされているバージョンを示します。サポートされているバージョンは次のとおりです。

  - Micro Focus COBOL V3.2 for UNIX
  - Micro Focus COBOL V4.0 for UNIX
  - Micro Focus Object COBOL V4.1 for UNIX
  - Server Express

コマンド行の書式の表記は次のとおりです。

- 斜体は、ユーザーの入力を示す一般的な名称を示します。
- 角かっこ [ ] は選択するオプションを示します。
- 中かっこ { } は、{ } の中からオプションを選択しなければならないことを示します。{ } にオプションが 1 つだけある場合は、そのオプションの繰り返しを示します。
- 角かっこ { } または中かっこ [ ] に続く省略記号 (. . .) は それらのオプションを繰り返すことができることを意味します。特に指定がない限り、その繰り返し回数に制限はありません。中かっこ [ ] の中で省略記号 (. . .) が使用された場合には、オプションをすべて省略できます。
- コマンド行がページ幅に収まらない場合には、次の行に継続されます。継続行はインデントされます。
- コマンド行のオプションは /option または -option として指定できます。

# 目次

# 第1章　はじめに

UNIX オプションにより、アプリケーション開発を UNIX システムから NetExpress にオフロードすることができます。つまり、NetExpress のツールを使用して PC 上でアプリケーションを作成し、準備ができたら、Micro Focus COBOL for UNIX がインストールされている UNIX システムに、作成したアプリケーションを展開できることを意味します。

---

備考: UNIX オプションを使用するには、必要なソフトウェアを UNIX システムにインストールし、その構成ファイルを変更する必要があります。これを行うには、UNIX システムへのスーパーユーザのアクセス権と、UNIX システムの構成に関連するスキルが必要です。特に、次のことに精通している必要があります。

- ftp の使用
- ファイルとディレクトリのコピー
- UNIX テキストファイルエディタの使用 (たとえば、vi、emacs)
- ファイルとディレクトリへのアクセス権の設定

これらのタスクに精通していない場合は、UNIX システム管理者に問い合わせてください。

また、UNIX システム上での DNS の使用による UNIX システムへの影響、および基本的なネットワークの問題についても、精通しておく必要があります。わからない場合は、UNIX システム管理者に問い合わせてください。

---

UNIX オプションを構成するものを次に示します。

- パブリッシャ- アプリケーションを UNIX システムにコピーし、UNIX システム上のビルド処理を制御します。パブリッシャのセットアップユーティリティにより、展開処理を制御することができます。たとえば、展開先の UNIX システム、アプリケーションのコピー先ディレクトリ、コピーファイルに使用するディレクトリなどを指定することができます。

  パブリッシャの使用法の詳細については、アプリケーションのパブリッシュの章を参照してください。パブリッシャの構成方法の詳細については、パブリッシャの設定の章を参照してください。

- インポートウィザード - Object COBOL for UNIX アプリケーションを NetExpress にインポートすることができます。 詳細については、NetExpress への UNIX アプリケーションのインポートの章を参照してください。
  Samba - 標準的な Windows のネットワークを使用して、UNIX システムへのリモートドライブアクセスを提供します。

NetExpress COBOL で使用できる構文には、COBOL for UNIX と互換性がないものもあります。NetExpress 上で WARNINGS (2) コンパイル指令を使ってコンパイルした場合には、コンパイラが互換性のない構文を見つけたときに警告メッセージを表示します。ただし、コンパイラが互換性のないすべての箇所にフラグを立てることはできません。互換性のない箇所の一部については、自分自身でチェックする必要があります。移植性の問題の章では、コンパイラが互換性のないものとしてフラグを立てることができる構文を一覧表示し、自分自身でチェックすべき互換性のない構文について説明します。

## 1.1 アプリケーションのパブリッシュについての概要

UNIX システムにアプリケーションを展開し、リビルドする処理は、「パブリッシュ」と呼ばれます。UNIX オプションには、パブリッシャと呼ばれる、パブリッシュ処理を処理するツールがあります。

---

備考: データアクセスウィザードを使用して NetExpress 上で生成した UNIX システムアプリケーションにパブリッシュすることはできません。

---

展開可能なアプリケーションを作成するための開発処理は次のとおりです。

1. NetExpress ツールを使用してアプリケーションを作成し、開発します。
2. WARNING(2) コンパイル指令を設定します。アプリケーションをコンパイルします。 COBOL for UNIX と互換性のない構文にはフラグが立てられます。必要に応じて構文を変更します。 COBOL for UNIX に問題を生じさせる可能性がある他の構文の詳細については、移植性の問題の章を参照してください。必要に応じて再コンパイルします。
3. エラーを発生することなくアプリケーションをビルドした場合は、次のことを行います。
    a. このプロジェクトに対して パブリッシャを設定します。詳細については、パブリッシャの設定の章を参照してください。通常は、プロジェクトに対してこの設定を一度だけ行います。
    b. アプリケーションをパブリッシュします。詳細については、アプリケーションのパブリッシュの章を参照してください。
4. アプリケーションを UNIX システムで実行します。ランタイムエラーが発生した場合には、NetExpress を使用して修正します。そのアプリケーションをリビルドし、再パブリッシュします。

NetExpress を使用して UNIX システム上の既存の COBOL アプリケーションを編集し、リビルドしたい場合には、そのアプリケーションを NetExpress にインポートする必要があります。このインポート方法については、NetExpress への UNIX アプリケーションのインポートの章を参照してください。

## 1.2 サーバーコントロールプログラム

サーバーコントロールプログラム (SCP) は、NetExpress と UNIX システムとの間の通信を可能にする機能の標準セットを付属しています。アプリケーションをパブリッシュするには、まず SCP を UNIX システムにインストールする必要があります。SCP のインストールについては、付録SCP と Samba のインストールを参照してください。

## 1.3 大文字と小文字の区別

UNIX システムにアプリケーションをパブリッシュする際に、リテラル (特にファイル名) の大文字と小文字の区別によって問題が発生する場合があります。一般的に、PC ではファイル名の大文字と小文字の区別は無視されます。たとえば、`filename1` は `FILENAME1` または `FiLeName1` と同等です。UNIX システムでは、`filename1` は `FILENAME1` とは異なるファイル名と見なされます。 その結果、次のようになります。

- NetExpress 開発環境では (ユーザーが読みやすいように) ファイル名を大文字にマップできますが、アプリケーションがビルドされるときに大文字と小文字がそのままである保証はありません。
- PC で作成されたプログラムの CALL 文と COPY 文で使用される文字では、大文字と小文字が混在する場合があります。この大文字と小文字の混在は PC では問題ありませんが、UNIX システムでは重要です。
- Microsoft や Novell の古いプロトコルを使用する PC ベースのファイルサーバーは、ファイル名の大文字と小文字の区別を無視します。新しいファイルサーバー (たとえば、Windows NT Server または Novell V4) では、大文字と小文字の混在をサポートする場合があります。

たとえば、myapp.cbl というプログラムを作成したとします。アプリケーションやプログラムを保存するために使用したオペレーティングシステム、または UNIX システムへの接続に使用しているネットワークプロトコルによって、実際のファイル名は、たとえば、myapp.cbl (Windows NT)、Myapp.cbl (Windows 95)、MYAPP.CBL (Novell のあるバージョン) となります。Windows ではこれらのファイル名はすべて同じであると見なされますが、UNIX ではすべて異なるファイルと見なされます。そのため、コード `call myapp.cbl` を含むプログラムを作成した場合には、呼び出しは Windows では期待どおり機能しますが、UNIX システムでは問題が発生します。パブリッシュされたプログラムのファイル名が myapp.cbl または Myapp.cbl である場合があるからです。

これらの問題を解決するには、ファイル名を特定の方法で処理するように パブリッシャを構成することができます。詳細については、パブリッシャの設定の章を参照してください。

# 第2章　移植性の問題

本章は、NetExpress コンパイラが、警告または通知エラーメッセージを使用してフラグを立てることのできる NetExpress と COBOL for UNIX との間の互換性がない構文について説明します。ただし、NetExpress コンパイラはすべての互換性のない箇所にフラグを立てることができないため、この章では UNIX システムにプログラムを移植する前に手動でチェックし、変更すべき箇所についても説明します。

Server Express をインストールずみの UNIX システムにパブリッシュする場合には、*「Server Express 以外」*とフラグの立っているセクションにおける移植性の問題は無視されます。

## 2.1 構文のチェック

NetExpress および COBOL for UNIX の両方の INTLEVEL 指令はデフォルトで 2 に設定されています。これは移植可能な中間コードを作成するために必要な値です。ただし、NetExpress COBOL の構文の中には、INTLEVEL"2" を指定してコンパイルした場合でも、UNIX システムで実行できない中間コードを生成するものもあります。

このため、WARNINGS"2" コンパイル指令を設定した場合には、NetExpress コンパイラは UNIX システムに移植されたときに問題を発生する可能性のある構文構造にフラグを立てます。

## 2.2 フラグを立てる構文

*Server Express 以外*
プログラムを INTLEVEL"2" および WARNINGS"2" 指令を指定して NetExpress でコンパイルした場合には、構文構造にフラグが立てられます。フラグは通知メッセージまたは警告メッセージの形式です。CHANGE-MESSAGE コンパイル指令を使用して、メッセージの重大度を情報または警告から重大なエラーに上げることができます。

- 160 バイトを超えるリテラル。COBOL for UNIX の制限は現在 160 バイトです。
- 呼び出し規約。NetExpress コンパイラは COB0L for UNIX コンパイラで受け付けない呼び出し規約を受け付けます。たとえば、NetExpress では呼び出し規約 256 と 512 の両方を受け付けますが、COBOL for UNIX では受け付けません。
- 空白文字を含むファイル名。
- それぞれ異なる呼び出し規約が指定されたエントリポイント。COBOL for UNIX では、最初のエントリポイントで残りのエントリポイントの呼び出し規約が設定されるものと想定されます。
- 4 バイトを超える値項目。COBOL for UNIX では、CALL 文で値で渡される最大バイト数は 4 バイトです。これより大きな値を渡そうとした場合には、オペレーティングシステムで障害が発生する場合があります。
- IS EXTERNAL DEFINITION および IS EXTERNAL REDEFINITION 句。これらの句は NetExpress でだけ

使用できます。

- 255 (COBOL for UNIX の最大数) を超えた場合のエントリポイントのパラメータ数。
- マルチスレッド構文。この構文は、COBOL for UNIX では使用できません。
- ソースプログラムで検出される Object COBOL サポート。

## 2.3 フラグを立てない構文

次の構文にはフラグを立てません。プログラム中に存在するかどうかをチェックする必要があります。

- UNIX では機能しないライブラリルーチン (すなわち、COBOL for UNIX のマニュアルで DOS、OS/2、または Windows だけと示されているものと、PC に特有のもの)。 たとえば、PC_FIND_DRIVES などの PC_library ルーチンは、COBOL for UNIX では使用できません。一部の call-by-number ライブラリルーチンは、NetExpress と COBOL for UNIX では異なります。たとえば、UNIX では x"AF"ルーチンを使用してマウス機能を制御することはできません。
- Panels2 への呼び出し。これらの呼び出しはサポートされていません。
- .dll ファイルへの明示的な呼び出し。
- UNIX での無効なファイル名。たとえば、拡張子 .dat をもつファイルあるいは空白文字やドライブ識別子を含むファイル名などです。
- リテラル中の埋め込み PC 文字（すなわち、印字不可能な文字）。
- ファイル記述中の COMP-5 データ項目。ファイルレコード記述で COMP-5 データ項目を指定した場合には、バイト順位が PC と UNIX システムでは異なる可能性があることに注意してください。
- プログラムがオブジェクト指向の構文を含む場合、COBOL for UNIX に対しては、ソースコードの先頭で $SET 文を使用するなどして、MFOO 指令を設定する必要があります。詳細については、次の節を参照してください。

COBOL for UNIX と NetExpress COBOL の違いについては、次の点についても気を付けてください。

- バイト順位。アプリケーションがプログラム間でパラメータの受け渡しをする場合には、システムがパラメータを渡す方法を認識する必要があります。UNIX システムの中には、PC で使用される逆のバイト順位ではなく、COBOL のバイト順位を使用するものもあります。このため、Intel チップを搭載した PC で動作するテスト済みのプログラムが、SPARC システムなどに移植したときに期待通りに動作しないことがあります。この問題は、COBOL と COBOL 以外のプログラムの間でパラメータを受け渡しする場合や、データを受け渡しするプログラムがそれぞれ異なるマシンで実行されている場合にだけ発生します。
- デフォルトのファイルタイプは PC と UNIX では異なります。

- 順ファイルに使用される区切り文字は、PC と UNIX では異なります。

## 2.4 LINKCOUNT 指令の使い方

*ServerExpress 以外*

COBOL for UNIX コンパイラがデータ項目に記憶スロットを割り当てる機構は、NetExpress コンパイラのものと異なります。このため、NetExpress を使用してコンパイルした正常に動作するプログラムを UNIX システムにパブリッシュすると、次のコンパイラエラーが発生する可能性があります。

```
0067 Please recompile using a larger value for the LINKCOUNT directive
```

プログラムを UNIX システムにパブリッシュしたときにこのコンパイルエラーが発生した場合には、COBOL ソースプログラムで $SET文 を使用して、エラーを発生したプログラムに LINKCOUNT 指令を設定する必要があります。

## 2.5 デバッグと .idy ファイル

.idy ファイルの形式は NetExpress と COBOL for UNIX 製品では異なります。そのため、NetExpress で作成された .idy ファイルは COBOL for UNIX では使用できません。

## 2.6 オブジェクト指向プログラム

*Server Express 以外*

Micro Focus COBOL for UNIX では、クラスと OO プログラムをチェックするために MFOO コンパイル指令を指定する必要があります。この指令によって Object COBOL の予約語がロードされます。NetExpress は MFOO 指令を受け付けますが、それを無視します。これは、すべての Object COBOL の予約語がデフォルトですでにロードされているからです。

OO プログラムまたはクラスを NetExpress で開発し、それを COBOL for UNIX を使用して再コンパイルするには、MFOO指令を指定する必要があります。この指定は、最良の方法として、ソースコードで $SET文 を使用して行います。

次のことに気を付けてください。

- ユーザークラスがクラスライブラリからデータとともにクラスを継承した場合には、これらのユーザークラスのコンパイルは UNIX では失敗します。これは、クラスライブラリが NetExpress のそれと異なるからです。データなしでクラスを継承することをお勧めします。
- クラス制御節の意味は、NetExpress 用に変更されました。この変更により、COBOL for UNIX で再コンパイルするときに問題が発生する場合があります。クラス制御エントリの形式は次のとおりです。

  ```
  class-control. logicalClassName is class "physicalFileName".
  ```

NetExpress では、logicalClassName の対象範囲はそれぞれクラスまたはそれが存在するプログラムです。これ以外の範囲には影響しません。Object COBOL の以前のバージョンでは、logicalClassName の対象範囲はグローバルな範囲です。複数のコンパイル単位で指定された logicalClassName に複数のエントリが存在する場合には、それらのエントリは一致する必要があります。一致しない場合には、ランタイムエラー 119 が発生します。

- COBOL for UNIX では、クラスはそのクラスがロードされるときに一度実行されるメソッド以外に手続き部をもつことができます。この構造は NetExpress では削除され、Initializeclass クラスメソッドで置き換えられました。 このメソッドはクラスが最初にロードされたときに、Object COBOL ランタイムシステムにより呼び出されます (結果として、COBOL for UNIX の手続き部を使用するのと同じ効果があります）。このため、Initializeclass メソッドは Micro Focus COBOL for UNIX では自動的に実行されません。

## 2.7 CGI アプリケーション

COBOL for UNIX は、CGI アプリケーションが要求するさまざまな構文およびランタイムサポートを、複数のレベルでサポートしています。たとえば、ACCCGI ランタイムサポートモジュールです。

UNIX オプションは、COBOL for UNIX 製品すべてについて、標準レベルの CGI 構文およびランタイムサポートを提供しています。COBOL for UNIX 製品が独自の構文またはランタイムサポートを提供している場合は、UNIX オプションは自動的にこれを使用します。

## 2.8 EUC についての留意点

日本語 EUC UNIX プラットフォームへ展開するために Windows 上で NetExpress を使用して開発するときは、文字体系はシフト JIS で開発することになり、ユーザが展開する文字体系と異なります。このため、文字コード変換の問題が生じます。また、EUC SBCS カタカナの場合は、データのサイズに関する問題も考慮する必要があります。

### 2.8.1 EUC サーバーからのインポート

EUC サーバーからインポートするときは、次の問題に注意する必要があります。

- EUC プラットフォームからインポートするときは、COBOL ソースコードは EUC からシフト JIS に変換されます。変換はソースファイルの内容に関係なく行われ、インポートウィザードを使用して行うことができます。

- 3 バイト EUC 文字コード (第 1 バイトは x"8F") は、未使用のシフト JIS 行、および拡張シフト JIS 文字を定義してマップする予定の行にマップされています。しかし、Micro Focus COBOL for UNIX は、3 バイト EUC 文字コードをサポートしません。既存のアプリケーションは、通常、3 バイト EUC 文字コードを含んでいないからです。

- 独自のユーザ定義漢字 (外字) を定義している場合は、Windows 上でこれらの外字を正しい値で定義する必

要があります。 変換の際に、EUC 行 85 ～ 94 は、シフト JIS 行 95 ～ 104 に変換されます。

- ユーザプログラムが非標準文字を含んでいる場合は、変換時に最初に非標準文字を検出したときに警告メッセージが出力されます。警告メッセージが出力された場合は、アプリケーションはコンパイルされない、または正しく表示されない可能性があります。変換されたアプリケーションの整合性をチェックしてください。

### 2.8.2 EUC サーバーへのパブリッシュ

EUC サーバーへパブリッシュするときは、次の点に注意してください。

- EUC サーバーへパブリッシュするときは、COBOL ソースコードは EUC からシフト JIS に変換されます。変換はソースファイルの内容に関係なく行われ、UNIX オプションのセットアップ画面で行うことができます。

- パブリッシュの際に、ファイル名とパス名はシフト JIS から EUC に変換されます。パブリッシュのオプションのテキスト検索と置換が、シフト JIS から EUC コードへの日本語文字の変換をサポートします。ただし、DBCS 文字範囲を指定して、検索および置換することはできません。

- ウィンドウズシステムは、拡張シフト JIS 文字 (第 1 バイトが x"FA"、x"FB" または x"FC") をサポートしている場合があります。しかし、拡張シフト JIS 文字は、UNIX システムでは使用できない可能性があります。 拡張シフト JIS 文字は、標準 EUC 文字範囲に変換されます (場合によっては、3 バイト EUC 文字も含みます)。

- 独自のユーザ定義漢字 (外字) を定義している場合は、UNIX 上でこれらの外字を正しい値で定義する必要があります。 変換の際に、シフト JIS 行 95 ～ 104 は、EUC 行 85 ～ 94 に変換されます。

- ユーザのプログラムが非標準文字を含んでいる場合は、変換時に最初に非標準文字を検出したときに警告メッセージが出力されます。警告メッセージが出力された場合は、アプリケーションはコンパイルされない、または正しく表示されない可能性があります。変換されたアプリケーションの整合性をチェックしてください。

### 2.8.3 Net Express がフラグをたてる EUC 文字

コンパイル指令 FLAGEUC は、EUC への変換時に移植性のないソースコード内で、特定のインスタンスにフラグをたてるために使用できます。 移植上の問題は、シフト JIS 16 進コードが明示的に指定されたとき、または特に半角カタカナが使用されたときに生じます。半角カタカナは、EUC では 2 バイト格納領域をしめるためです。

次の場合にフラグをたてます。

- 定数の中に半角カタカナ (1 バイト）がある場合すべて。

- カタカナまたは DBCS 領域内に、16 進値を含む 16 進定数すべて。

- UNIX へのアップロード時にエラーを作成するソース内に、半角カタカナがある場合すべて。

拡張シフト JIS DBCS 文字またはユーザ定義 JIS DBCS 文字を使用していてフラグをたてていない場合は、これらの文字を使用する前に UNIX プラットフォームがこれらの文字をサポートしていることを確認してください。

# 第3章　パブリッシャの設定

パブリッシャを使用してアプリケーションをパブリッシュするには、次のことを行う必要があります。

1. アプリケーション用の NetExpress プロジェクトを作成します。まったく新規にプロジェクトを作成するか、あるいは既存のソースファイルに基づいてプロジェクトを作成します。アプリケーションの作成とプロジェクトの使用法については、NetExpress のヘルプを参照してください。

   既存の UNIX アプリケーションに基づいてアプリケーションを作成したい場合には、その既存のアプリケーションを NetExpress にインポートします。詳細については、NetExpress への UNIX アプリケーションのインポートの章を参照してください。

2. UNIX システムを設定します。

   - ターゲットの UNIX システムで パブリッシャのログインを選択します。
   - .rhosts ファイルを構成して、パスワードを必要としないログインを有効化します。

   詳細については、付録 A「SCP と Samba のインストール」を参照してください。

プロジェクトを作成し、UNIX システムを設定した場合には、「パブリッシャ セットアップ」ダイアログを使用してプロジェクトに対して パブリッシャを構成することができます。

---

備考: 次に示す「パブリッシャ セットアップ」ダイアログには、UNIX システムのディレクトリやファイル情報を入力するためのいくつかのテキスト領域があります。たとえば、ビルド ディレクトリ、copyfile ディレクトリなどです。これらのファイル名とディレクトリには絶対パス (スラッシュ (/) で始まる) と相対パスのどちらでも使用できます。相対パスは指定したユーザーのホームディレクトリに関連しています。たとえば、ユーザー `foo` のホームディレクトリが `/usr/foo` の場合には、そのユーザーは ビルド ディレクトリを `/usr/foo/mybuilddir` または `mybuilddir` として指定できます。どちらの指定も同じです。

環境変数も、サーバーに渡されるパスでのワイルドカードの展開も指定できないことに注意してください。

---

次の説明では新規設定を想定しています。「パブリッシャ セットアップ」ダイアログを使用して、既存の構成を編集することもできます。その場合、以前に設定された値がデフォルトとして表示されます。

1. IDE の [UNIX] メニューをクリックします。
2. [セットアップ] をクリックします。次のダイアログボックスが表示されます。


タブをクリックして、次の設定を行います。

- プロジェクト
- サーバー
- その他の UNIX オプションの詳細

タブをクリックすると、各構成の詳細を入力するためのフォームが表示されます。以前に構成の設定の詳細を指定した場合には、その構成の値が表示されます。

各プロジェクトに対して、プロジェクトのパブリッシュ先のさまざまなサーバーを設定することができます。各サーバーに対して、異なる構成情報を指定することができます。

## 3.1 プロジェクトの詳細設定

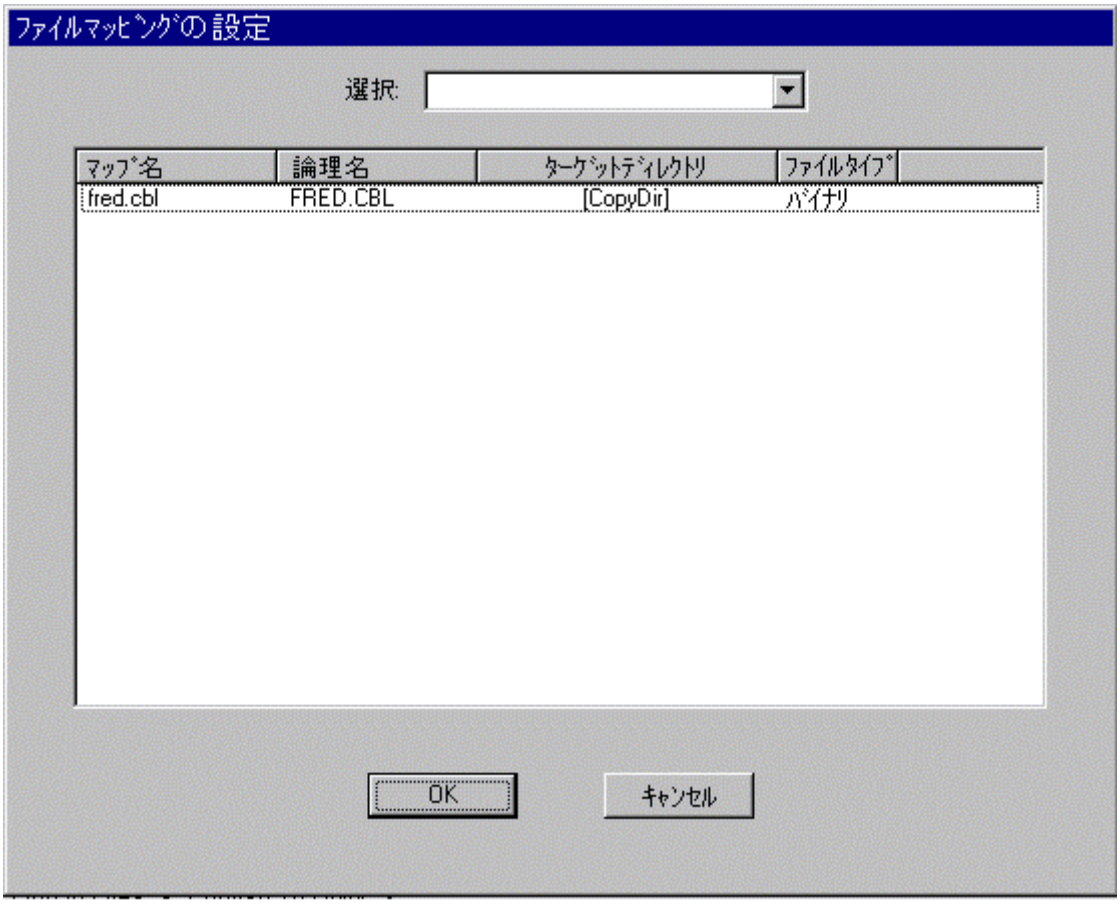
プロジェクトの設定情報を入力するには、必要に応じて「UNIX オプションセットアップ」ダイアログの [プロジェクト]タブ (デフォルトのタブ) をクリックします。

最初のフィールドには、現在アクティブなサーバーの構成が表示されます。サーバーが現在選択されていない場合には、このフィールドには 選択されていません が表示されます。 UNIX システムに CGI アプリケーションをパブリッシュしたい場合は、[CGI サポートを使う] をチェックします。

ボタンを使用して、さらに詳細を設定することができます。これらのボタンについては、次の節で説明します。

### 3.1.1 ファイル名のマッピング

UNIX システムへパブリッシュされるときのファイル名、位置、各ファイルの種類を指定するには、[ファイル名マッピング] ボタンを使用します。CALL 文などの文で入力したファイル名と一致させるために、このマッピング方法の指定が必要になる場合があります。




列 「マップ名」 は、UNIX へコピーされたときの実際のファイル名を大文字・小文字を区別して指定します。

デフォルト値は、Windows ファイルシステムが決定するファイル名となります。

列 「論理名」 は、NetExpress で表示されたファイル名を表示します。論理名は常に大文字であり、ユーザが編集することはできません。

ファイルが UNIX システムにコピーされるときに使用するマップドファイルの名前を入力することができます。 たとえば、プログラムに呼び出し文 `call "MYapp"` があるとします。ファイル名マッピング手続きにより、Windows で Myapp.cbl として認識されているファイルが UNIX システム上では MYapp.cbl として作成されるように明示的に指定することができます。

ビルドプロセスの一部としてマップされた名前で作成されたすべてのファイルでは、大文字と小文字が同じようにマップされます。 たとえば、MYapp.cbl は MYapp.int となり、MYAPP.cbl は MYAPP.int となります。

「ファイルマッピング構成」ダイアログを使用して、ファイル名を変更することはできません。たとえば、Myapp.cbl を Myapp2.cbl に変更するために Myapp.cblを変更することはできません。

ファイル名のマッピングと大文字と小文字の区別については、*はじめに*の章の 「大文字と小文字の区別」 の節を参照してください。

列 「ターゲットディレクトリ」 は、ファイルのコピー先の論理ディレクトリを指定します。 論理ディレクトリは、「論理ディレクトリ」ダイアログで定義されます。

列 「ファイルタイプ」 は、ファイルを UNIX サーバーにバイナリファイルとして転送するか、テキストファイルとして転送するかを指定します。 新規ファイルのデフォルトは、テキストです。

ファイルをテキストとして指定した場合は、次のようになります。

- 改行文字はすべて、PC 形式の文字 CR/LF から、UNIX 形式の文字 LF に変換されます。
- どのような検索・置換操作に対しても正しいファイルとなります。
- 「サーバー設定」ダイアログで「EUC サーバー」チェックボックスを可能にした場合は、EUC 文字エンコードに変換されます。.

---

備考: プロジェクト内のバイナリファイルを、絶対にテキストファイルとして定義しないでください。 プロジェクト内のバイナリファイルをテキストファイルとして定義した場合は、サーバー上のファイルは破壊されます。

---

[選択] プルダウンメニューは、事前に定義したパターンに一致する一連のファイルを簡単に選択することができます。 たとえば、COBOL ファイルすべて (*.cbl)、またはユーザ自身が選択したパターンです。

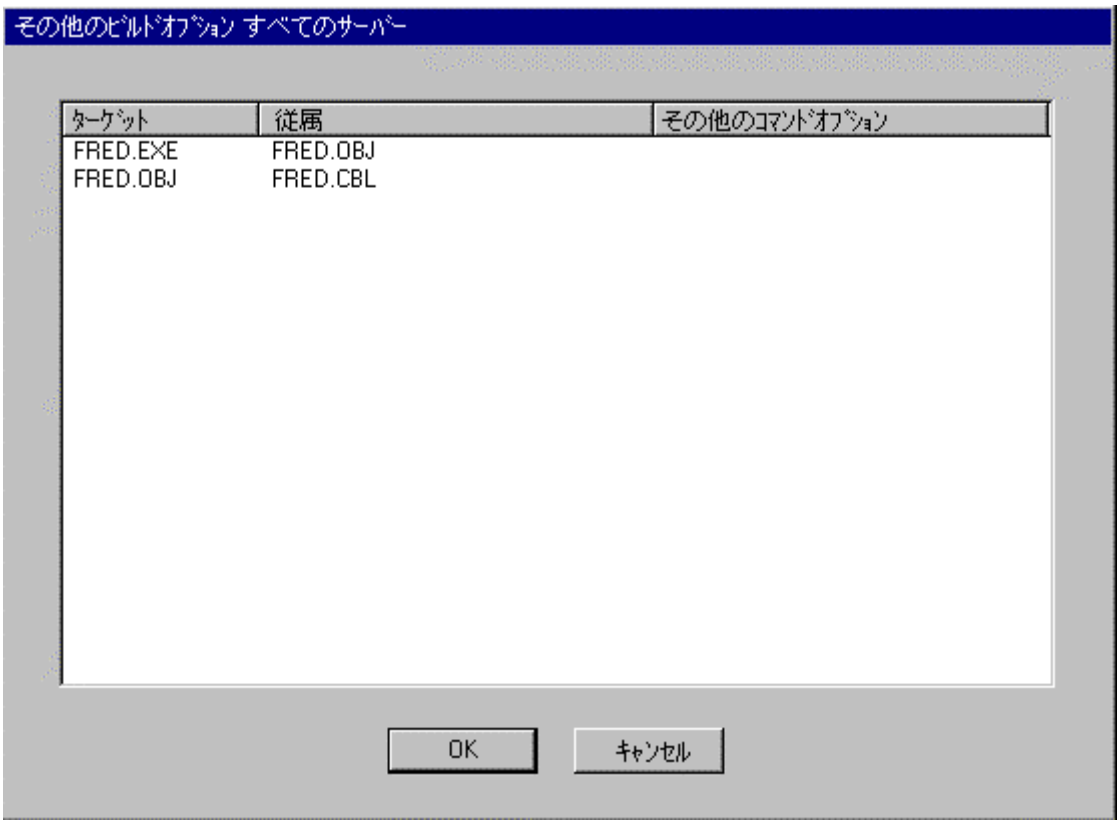
ファイルマッピング情報項目を編集する方法は、2 通りあります。

- 編集したい行を選択します。 選択した行内で編集したいフィールドをクリックします。 ファイル名などの項目については、編集フィールドが表示されます。 他のフィールド（ファイルタイプなど）については、有効な値のプルダウンリストが表示されます。
- [選択] プルダウンメニュー、または Windows の標準的な複数選択の方法を使用して、複数の項目を選択します。 編集したい行で右クリックすると、その列の有効な操作を含むコンテキストセンシティブメニューが表示されます。 選択したファイルすべてに操作が適用されます。

### 3.1.2 追加ビルドオプションの設定

選択したプロジェクトに対して、[追加ビルドオプション] ボタンをクリックすることによって、プロジェクトをビルドするときに cob コマンド行に追加されるコンパイル指令を指定することができます。指定された指令は、現在

のプロジェクトに対して定義されたすべてのサーバーに適用されます。必要な指令を入力するためのダイアログが表示されます。




列「ターゲット」 は、作成される論理名です。たとえば、CGIPRG1.INTです。

列「依存」 は、ターゲットを作成するために使用される論理名をリストします。たとえば、CGIPRG1.CBLです。

列「指令」 は、追加する cob コマンド行オプションを指定するフィールドです。 設定を編集するには、編集したい行を選択し、選択した行で列「指令」をクリックします。 または、編集した行でマウスを右クリックし、メニュー項目 [オプションを編集] を選択します。

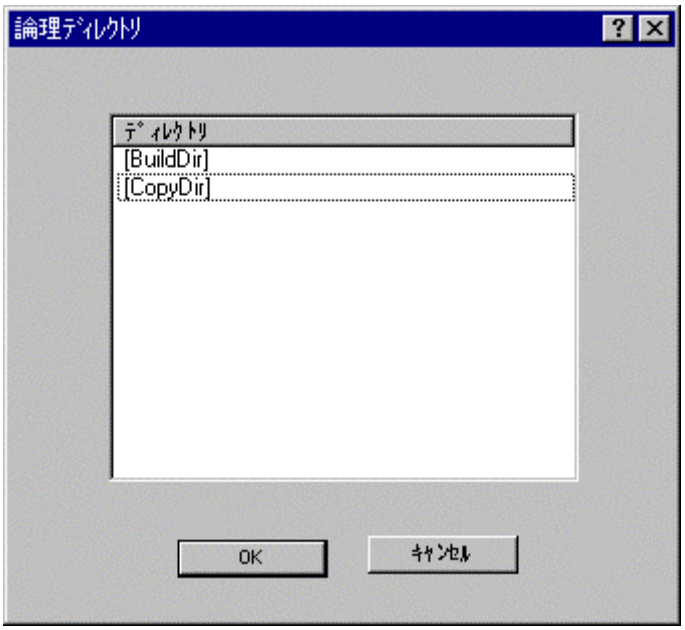
---

備考: ユーザが入力した指令は、パブリッシャによってチェックされません。したがって、変更されることなく cob コマンド行に渡されます。

---

### 3.1.3 論理ディレクトリの設定

プロジェクトに対して論理ディレクトリ名を追加したり、削除することができます。[プロジェクト] タブの [論理ディレクトリ] ボタンをクリックします。 ダイアログボックスが表示され、論理ディレクトリを入力することができます。




2 つの特別な論理ディレクトリが常時、表示されていますが、削除することはできません。

- [BuildDir] は、ビルド操作すべての基本となるディレクトリです。また、複数のユーザが同じ領域にアクセスしないようにロックされているディレクトリです。 NetExpress V2.0 Unix Option で使用可能な 「ビルドディレクトリ」 設定に対応しています。
- [CopyDir] は、NetExpress V2.0 UNIX Option の「サーバーコピーブックディレクトリ」設定と互換性があります。

リストコントロール内で右クリックすることによって、論理ディレクトリを削除したり作成したりできます。 論理ディレクトリ名はフリーフォーマットテキストであり、有効なテキスト文字を含むことができます。

論理ディレクトリはプロジェクト全体の設定です。 論理ディレクトリは、論理的にファイルをコピーする場所を指定するために、「ファイル名マッピング」ダイアログで使用されます。また、「サーバ設定」ダイアログの [ディレクトリ指定] タブ内で、サーバー間の基礎となる実際の値が割当てられます。

---

備考: 論理ディレクトリを削除した場合は、この値を使用していたマッピング項目は、代わりに [BuildDir] を使用するように更新されます。

---

### 3.1.4 特定のプロジェクトの探索/置換パターンの設定

特定のプロジェクトのパターンを編集し、置換するには、[検索/置換] ボタンをクリックします。現在の探索/置換パターンのリストと適用されるファイルが表示されます。次に例を示します。


列「探索パターン」は、正規表現を指定します。正規表現については、付録「正規表現」で定義しています。

列「置換パターン」は、探索パターンが成功したときに置換するテキストを指定します。 置換メタキャラクタについては、付録「正規表現」で定義しています。

---

備考: ファイル名が列「対象ファイル指定パターン」で定義したファイルに一致しても、「ファイル マッピング設定」ダイアログ内でテキストファイルとして定義されていなければ、処理されません。

---

Publish 操作中は、リストボックスの表示順にパターンが実行されます。 パターンの順番を変更するには、行を選択し、希望する位置へパターンをドラッグ & ドロップします。

パターンを編集するには、カーソルでそのパターンを選択し、左クリックします。 検索パターンを編集する方法の例を次に示します。

1. マウスを使用してパターンを選択します。
2. 左クリックします。
3. パターンを編集します。

右クリックすることで、パターンを編集したり、新しいパターンを作成することもできます。新しいパターンを作成し、リストの最後にそのパターンを追加する方法を次に示します。

1. 画面の空白領域を右クリックします。次のダイアログが表示されます。


2. ダイアログに新しいパターンを入力します。

3. [OK] をクリックします。

[検索/置換設定] リストのパターンリストの最後に新しいパターンが追加されます。

新しいパターンを作成し、リストの特定の場所にそのパターンを追加する方法を次に示します。

1. 新しいパターンを追加したい行の前の行のフィールドの 1 つを右クリックします。新しいパターンを追加するか、または現在のパターンを編集するかどうかを尋ねられます。

2. ダイアログに新しいパターンを入力します。

3. [OK] をクリックします。

右マウスボタンを使用してパターンを編集する方法を次に示します。

1. 変更したいパターンを含む行を右クリックします。ダイアログボックスのフィールドには、その行で定義されたパターンの指定とファイルの指定がすでに設定されています。

2. 必要に応じてパターンの指定とファイルの指定を編集します。

3. [OK] をクリックします。

## 3.2 サーバーの詳細設定

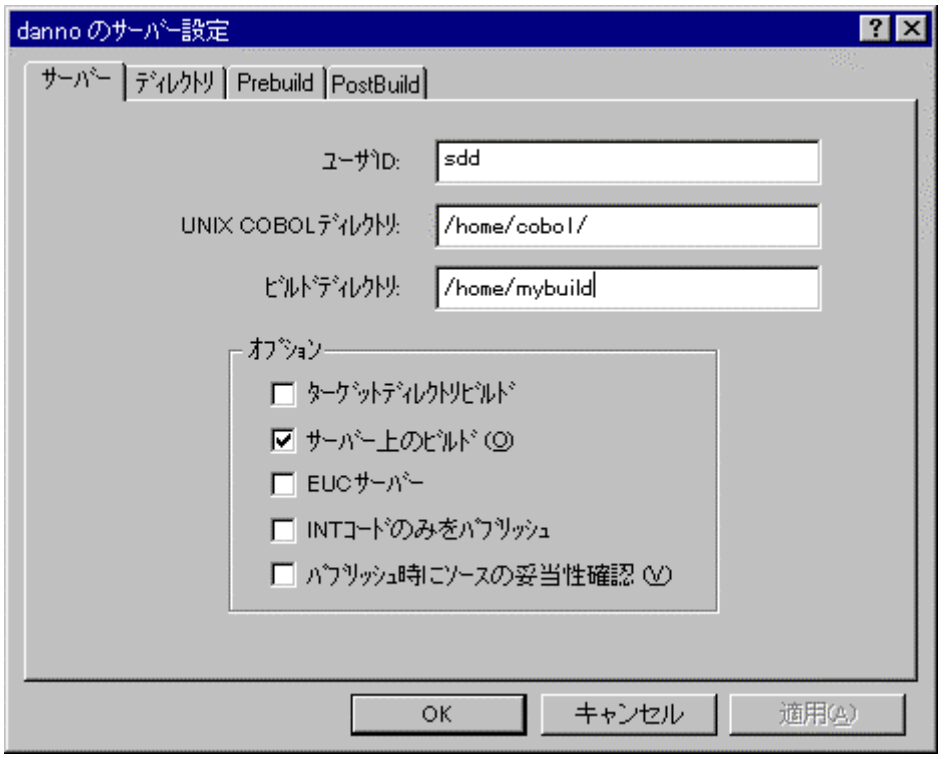
サーバーの設定情報を入力するには、[サーバー] タブをクリックします。次のフォームが表示されます。


リストボックスは、現在定義されているサーバーを表示します。これらの 1 つを選択した場合は、このタブのボタンを使用して行った変更が選択したサーバーに適用されます。新規サーバー エントリで、新しいサーバーを定義することができます。 このエントリを選択すると、新しいサーバー名を指定するためのダイアログが表示されます。

このタブのボタンを使用して、さらに詳細に設定できます。これらのボタンについては、次の節で説明します。

### 3.2.1 サーバー設定

[サーバー設定] ボタンをクリックすると、次のタブ付きのダイアログが表示されます。




タブをクリックして、次の設定を行います。

- サーバー
- サーバーのディレクトリ
- Prebuild コマンド
- Postbuild コマンド

### 3.2.1.1 サーバーの詳細設定

サーバーの詳細を設定するには、必要に応じて 「サーバー設定」 ダイアログの [サーバー] タブ (デフォルトのタブ) をクリックします。ここで指定する設定は、指定したサーバーにだけ適用されます。

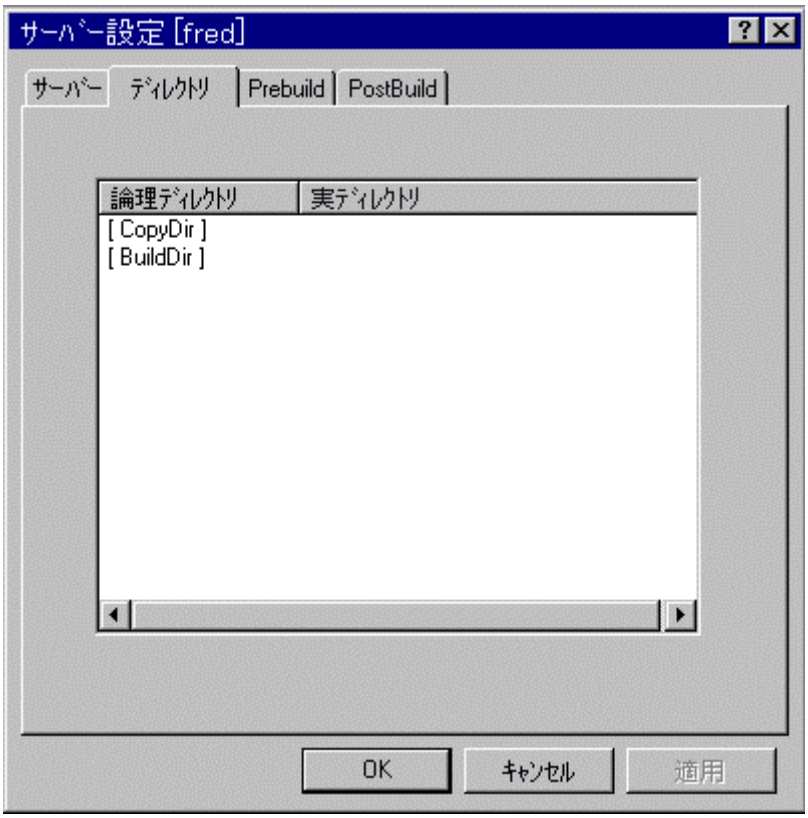
次の詳細を入力してください。

| | |
|---|---|
| ユーザ Id | 上記で指定したサーバーへの接続に使用するログイン識別子。デフォルトは、Windows にログインするときに使用するユーザー ID です。 |
| UNIX COBOL ディレクトリ | Micro Focus COBOL for UNIX が入っている UNIX システムのディレクトリ。このフィールドに詳細を入力すると、「UNIX オプションセットアップ」 ダイアログ上の[サーバーの容量] ボタンが有効化されます。 |

| | |
|---|---|
| ビルドディレクトリ | プロジェクトのパブリッシュ先の UNIX システムのディレクトリ。このフィールドに詳細を入力すると、「UNIX オプションセットアップ」ダイアログ内の [サーバーの排他制御] ボタンが有効化されます。 |
| ターゲット ディレクトリ ビルド | アプリケーションを NetExpress のターゲット型で指定されたサブディレクトリに組み込みたい場合は、これをチェックします。たとえば、Debug などです。このディレクトリは ビルド ディレクトリ で指定されたディレクトリ下に作成されます。 |
| サーバー上でビルド | ファイルをリビルドせずに UNIX システムにコピーしたい場合には、チェックをはずします。 |
| EUC サーバー | UNIX システム上で EUC サポートを有効化したい場合はこれをチェックします。 |
| INT コードのみをパブリッシュ | ソースコードではなく、中間コードからプロジェクトをビルドしたい場合にはこれをチェックします。 |
| パブリッシュ時にソースを検証 | 各パブリッシュ操作においてソースを検証したい場合に、これをクリックします。UNIX システム上で変更されたファイルを走査します。 |

#### 3.2.1.2 ディレクトリの指定

ディレクトリの詳細を設定するには、「サーバー設定」 ダイアログの [ディレクトリ] タブをクリックします。次のダイアログが表示されます。




リストボックスが表示されます。このリストボックスには、([プロジェクト] タブから) 「論理ディレクトリ」ダイアログ で定義した論理ディレクトリと現在のサーバー上の実際の物理ディレクトリが含まれます。 [BuildDir] の値は、[サーバー] タブで入力した値と常に一致します。どちらかの値を変更すると、自動的にもう一方へ反映されます。

どの論理ディレクトリにも値がない場合は、最初に使用された時点での現在の [BuildDir] の値が自動的に割り当てられます。 いったん、この値が割り当てられると、[BuildDir] を変更しても他の論理ディレクトリ値には影響しません。

### 3.2.1.3 Prebuild コマンドと Postbuild コマンドの設定

プロジェクトがビルドされる前後に実行する 1 つまたは一連のコマンドを定義することができます。Prebuild コマンドを定義するには、「サーバー設定」 ダイアログの [Prebuild] タブをクリックします。PostBuildコマンドを定義するには、[PostBuild] タブをクリックします。

---

備考: UNIX の `make` プログラムの動作に従って、各コマンド行は並行するシェル環境で実行されます。 したがって、値をある 1 行に設定し、後でプリビルドまたはポストビルドのコマンドシーケンス中にこの値を読むことはできません。 `for` ループなどの処理を含め、このような処理をする必要がある場合は、サーバー上にスクリプトを作成するか、サブシェル内でコマンドを実行するかする必要があります。

---

### 3.2.2 サーバー名の削除

サーバーリストから現在、選択しているサーバーを削除するには、[削除] ボタンを使用します。

サーバー名を削除する手順は、次のとおりです。

1. 「現在のサーバー」ボックス内のプルダウンメニューリストから、削除したいサーバー名を選択します。
2. [削除] ボタンをクリックします。削除してよいかどうか確認されます。

### 3.2.3 サーバーロックの変更

[サーバーの排他制御] ボタンをクリックして、UNIX システムの ビルド ディレクトリのロック、およびロックの解除を指定することができます。現在のロック状態を示すダイアログが表示され、ビルド ディレクトリをロックまたはロック解除することができます。

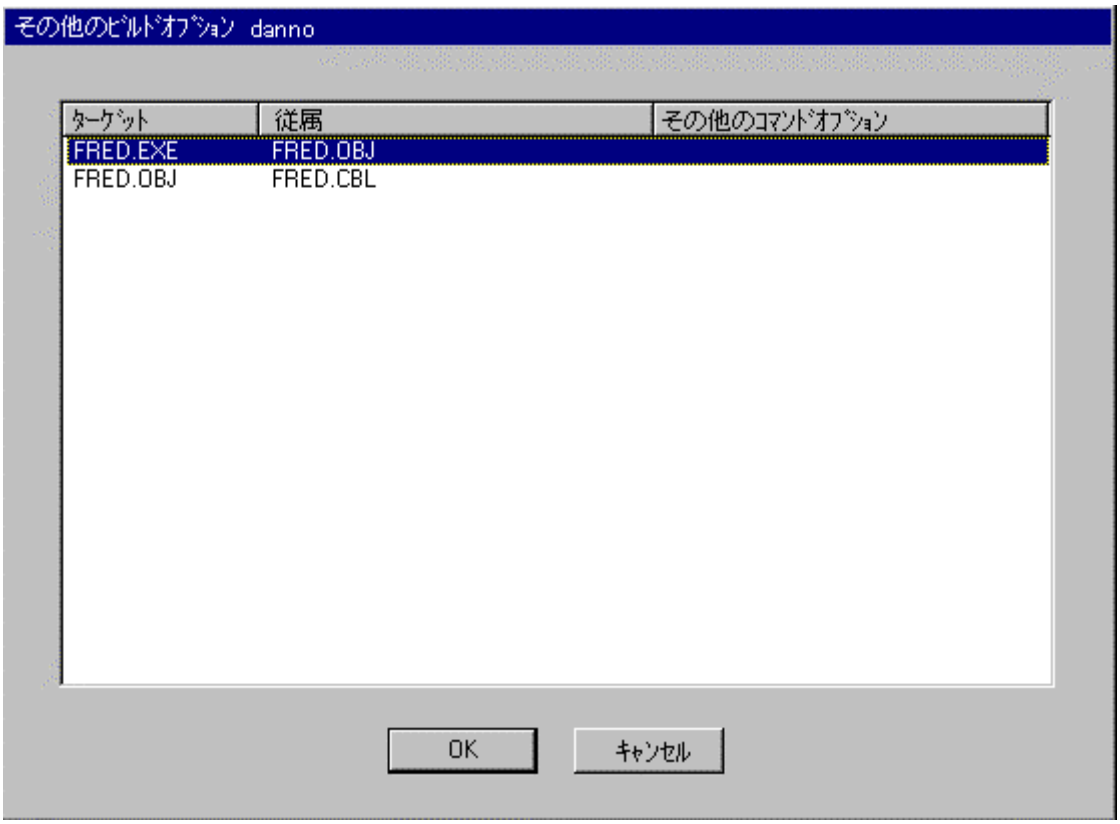
### 3.2.4 サーバー設定の検証

[サーバー設定の検証] をクリックすると、次のことができます。

- UNIX システム上 の Object COBOL for UNIX に関する情報を表示します。たとえば、Object COBOL for UNIX が共有オブジェクトファイルをサポートするかどうかなどです。
- 「セットアップ」ダイアログで入力されたエントリを検証します。[パブリッシュ] オプションを正しく機能させるには、報告されたエラーを修正してください。

### 3.2.5 追加ビルドオプションの設定

[追加ビルドオプション] ボタンをクリックすることで、プロジェクトをビルドするときに cob コマンド行に追加される指令を、選択したサーバーに対して指定することができます。 ここで設定された指令は、現在のサーバーに適用されます。 必要な指令を入力するためのダイアログが表示されます。




列 [ターゲット] は、作成されている論理名です。例: CGIPRG1.INT

列 「依存」は、ターゲットを作成するために使用されている論理名をリストします。例: CGIPRG1.CBL

列 「指令」は、追加 cob コマンド行オプションを指定するフィールドです。設定を編集するには、編集したい行を選択し、選択した行で列「指令」をクリックします。 または、編集したい行の上でマウスを右クリックし、メニュー項目 [オプションを編集] を選択します。

---

備考: ユーザが入力した指令は、パブリッシャによってチェックされず、変更されずに cob コマンド行に渡されます。

---

### 3.2.6 特定のサーバーの探索/置換パターンの設定

選択したサーバーに対して、テキストファイルがサーバーにパブリッシュされるときに処理されるファイル名のパターンと正規表現を指定することができます。 このパターンはリストとして作成され、そのリストに追加された順番に従って実行されます。

パターンを編集し、置換するには、[検索 / 置換設定] ボタンをクリックします。現在の探索 / 置換パターンのリストと適用されるファイルが表示されます。次に例を示します。


列「探索パターン」は、正規表現を指定します。正規表現は、付録「正規表現」で定義されます。

列 「置換パターン」は、検索パターンが成功したときに置換するテキストを指定します。 置換メタキャラクタは、付録「正規表現」で定義されます。

---

備考: ファイル名が、列「対象ファイル指定パターン」に定義したファイルに一致した場合は、「ファイル マッピング設定」ダイアログ内でテキストファイルとして定義されているときだけ、処理されます。

---

パブリッシュ操作中は、リストボックスの表示順にパターンが実行されます。 パターンの順番を変更するには、行を選択し、希望する位置へパターンをドラッグ & ドロップします。

パターンを編集するには、カーソルでそのパターンを選択し、左クリックします。 探索パターンを編集する方法の例を次に示します。

1. マウスを使用してパターンを選択します。
2. 左クリックします。
3. パターンを編集します。

右クリックすることで、パターンを編集したり、新しいパターンを作成することもできます。新しいパターンを作成し、リストの最後にそのパターンを追加する方法を次に示します。

1. 画面の空白領域を右クリックします。次のダイアログが表示されます。


2. ダイアログに新しいパターンを入力します。

3. [OK] をクリックします。

[検索/置換設定] リストのパターンのリストの最後に新しいパターンが追加されます。

新しいパターンを作成し、リストの特定の場所にそのパターンを追加する方法を次に示します。

1. 新しいパターンを追加したい行の前の行のフィールドの 1 つを右クリックします。新しいパターンを追加するか、または現在のパターンを編集するかどうかを尋ねられます。

2. ダイアログに新しいパターンを入力します

3. [OK] をクリックします。

右マウスボタンを使用してパターンを編集する方法を次に示します。

1. 変更したいパターンを含む行を右クリックします。ダイアログボックスのフィールドには、その行で定義されたパターンの指定とファイルの指定がすでに設定されています。

2. 必要に応じてパターンの指定とファイルの指定を編集します。

3. [OK] をクリックします。

### 3.2.7 ソースコードの検証

PC のバージョンと UNIX システムのバージョンの間でソースコードのステータスを検証することができます。この検証を行うには、[ソースの検証] ボタンをクリックします。リストボックスが表示されます。


列「ステータス」は、ファイルのステータスを表示します。 スタータスには次のものがあります。

| | |
|---|---|
| OK | PC 上のファイルと UNIX 上のファイルのパブリッシュ履歴は、一致します。 |
| 次のパブリッシュでコピー | ファイルはこれまでパブリッシュされたことがないか、次のパブリッシュでコピーしたいファイルを指定しました。 |
| サーバーのファイルが古い | PC 上のファイルが最後にパブリッシュされてから変更されています |
| サーバーのファイル修正済 | UNIX システム上のファイルが最後にパブリッシュされてから変更されています。 |
| サーバーのファイルが見つかりません | サーバー上でソースファイルが見つかりませんでした。 |

行を右クリックすると、コンテキストメニューが可能になり、次にパブリッシュするときにファイルをコピーできます。

## 3.3 その他の詳細設定

「セットアップ」ダイアログの [一般設定] タブを使用して、プロジェクト固有のものでもサーバー固有のものでもない、その他の詳細を設定することができます。このタブは、PC から UNIX システムへのリモート端末アクセス用に使用される端末エミュレータを指定するために使用されます。

[一般設定] タブをクリックすると、次のダイアログが表示されます。


デフォルトでは、UNIX オプションは Windows の telnetユーティリティを使用します。別の端末エミュレータを使用したい場合には、次に示す手順に従ってください。

1. エントリフィールドで、選択したい端末エミュレータのパス名と実行可能ファイル名を指定します。必要な実行可能ファイルの場所またはファイル名が確かでない場合には、[参照] ボタンを使用します。

   telnet プログラムが存在する場合は、デフォルトで telnet が選択されます。

# 第4章　アプリケーションのパブリッシュ

UNIX オプションにより、NetExpress でアプリケーションを作成し、UNIX システムにその作成したアプリケーションをパブリッシュすることができます。そのため、アプリケーション開発が UNIX 環境からオフロードされ、NetExpress のツールと機能を使用することができます。アプリケーションのパブリッシュは、NetExpress パブリッシャにより処理されます。すべてのソースコード管理は PC 上で行われ、パブリッシャに透過的です。たとえば、NetExpress に付属の PVCS ソースコード管理パッケージを使用することができます。

## 4.1 パブリッシャ

パブリッシャへのアクセスは NetExpress 統合開発環境 (IDE) の [UNIX] メニューで行われます。UNIX オプションがインストールされると、[UNIX] メニューは、アプリケーションをパブリッシュを可能にする次のオプションを提供します。

- パブリッシュ - アプリケーション内の修正したプログラムを UNIX システムにパブリッシュします。
- すべてをパブリッシュ - アプリケーション内のすべてのプログラムをパブリッシュします。
- パブリッシュ中止 - 可能な場合には現在のパブリッシュ操作を停止します。
- セットアップ - 作業に適した パブリッシャの設定を有効化します。パブリッシャの設定の章を参照してください。

---

備考: NetExpress データアクセスウィザードを使用して生成した UNIX システムアプリケーションにはパブリッシュできません。

---

## 4.2 パブリッシャの使い方

UNIX システムにアプリケーションをパブリッシュするには、[UNIX] をクリックします。[パブリッシュ] をクリックして、最後にパブリッシュしてから修正したソースファイルだけを指定したサーバーにパブリッシュするか、[すべてをパブリッシュ] をクリックしてプロジェクト内のすべてのファイルをパブリッシュします。

「サーバー設定」ダイアログで指定したディレクトリがサーバー上にみつからない場合は、パブリッシャにディレクトリを作成させるかどうかきいてきます。 ディレクトリを作成したくない場合、またはディレクトリの作成に失敗した場合は、パブリッシュ操作は失敗します。 サーバー設定をチェックし、ディレクトリを手動で作成する必要があります。

「サーバー設定」ダイアログで [パブリッシュ時にソースを検証] をチェックした場合は、サーバー上のファイルが

いっさい変更されていないことをチェックするために、プロジェクト内のファイルすべてが検証されます。 変更されたファイルがあった場合は通知があり、パブリッシュを継続するか、または停止するかのオプションが表示されます。

---

備考: ソース検証オプションは、ソースファイル上で CRC32 チェックを実行します。CRC32 チェックは信頼性の高い変更を検出しますが、ファイルをすべて読込む必要があります。 これは時間のかかるプロセスであり、UNIX システム上でファイル変更を行った可能性がある環境で操作している場合だけ、このオプションを可能にしてください。

---

プロジェクトに関連した設定を変更し、[パブリッシュ] をクリックした場合は、新しい設定を有効化するためにすべてのファイルがパブリッシュされます。

パブリッシュ操作の進捗は、NetExpress IDE の「アウトプット」ウィンドウでモニタされます。

## 4.3 CGI プログラムのパブリッシュ

CGI プログラムのパブリッシュには気を付けなければならない手順が他にもいくつかあります。

---

備考: パブリッシャは UNIX システムへの CGI プログラムのパブリッシュを処理することができます。ただし、UNIX システムで CGI アプリケーションを展開するときに考慮する必要がある他の要素について注意が必要です。インターネットアプリケーション (PIpubb03.htm) ブックの次の節をお読みください。

- UNIX システムへの CGI アプリケーションの展開を準備する方法については、『展開手順ガイド (UNIX サーバー)』 を参照してください
- UNIX への CGI アプリケーションのパブリッシュの詳細については、UNIX へのアプリケーションのパブリッシュ を参照してください。

---

次に示す手順は CGI アプリケーションのパブリッシュの概要を示します。

1. サーバーの設定を指定するには、「UNIX セットアップ」ダイアログ ([UNIX セットアップ] をクリック) を使用します。 「UNIX オプションセットアップ」ダイアログ内で [CGI サポート] をクリックし、アプリケーションが CGI アプリケーションであることを指定する必要があります。

   CGI プログラムの中で、Form Designer が大文字でフォームのファイル名を指定するため、ファイル名のマッピングが正しく構成されていることを確認してください。 ACCCGI モジュールは小文字の .htm 拡張子をもつファイルだけを検索できます。大文字の .HTM または小文字の .html 拡張子をもつファイルは検索できません。

2. CGI アプリケーションの NetExpress プロジェクト定義では、アプリケーションがシステム実行可能 (.exe) ファイルとしてコンパイルされることを確認してください。

3. [UNIX] メニューで [パブリッシュ] をクリックします。アプリケーションファイル、必要なプリプロセッサ、ランタイムサポートモジュールがリモートシステムにコピーされ、ビルドプロセスが修正されます。

   HTML ファイルがソースプールにある場合には、これらのファイルも同様に UNIX システムに転送されます。HTML ファイルを (手動で、または パブリッシャの検索 / 置換機能を使用して) 修正し、CGI トリガプログラム（次の手順を参照）の URL を指すようにする必要があります。トリガへのパスは web サーバーの構成により異なります。

4. CGI を実行するには、UNIX システム上でトリガプログラム (通常はシェルスクリプト）を作成します。 パブリッシャが作成した mfenv.sh シェルスクリプトを使用することができます。この場合は、完全なトリガは次のようになります。

   ```
   #!/bin/sh
   ./mfenv.sh cgiapp
   ```

   トリガを保存します。

   ---

   備考: Server Express にパブリッシュする場合には、この手順は必要ありません。

   ---

5. トリガファイル、すべてのアプリケーションおよび HTML ファイルを UNIX システムの指定されたディレクトリにコピーします。 通常の HTML ファイルは、web サーバーの適切な場所に置く必要があります。CGI 実行可能ファイルとサポートファイルは、web サーバーで実行可能である必要があるため、実行可能な場所に置かなければなりません（NCSA または Apache などのほとんどの UNIX web サーバーの場合は、実行可能な場所は ScriptAlias 構成パラメータにより制御されます）。

   次の場合には、ファイルは正しい場所に自動的にコピーできます。

   - 追加論理ディレクトリを定義する場合
   - ファイルをコピーするためのポストビルドコマンドを指定する場合

6. すべてのユーザーがファイルを読み取ることができるようにします。たとえば、ファイルのアクセス権を次のように変更します。`chmod +r *.htm`

CGI ディレクトリにビルドディレクトリの別名を指定することができます。この別名の指定は、迅速な変更を行うのに便利です。これは、ディレクトリ間でファイルをコピーする必要がないからです。

「セットアップ」ダイアログの 「PostBuild コマンド」フィールドにコマンドを入力して、ファイルを正しい CGI デ

ィレクトリにコピーすることができます。

CGI アプリケーションをパブリッシュするときに、ランタイムサポートモジュール ACCCGI.int がビルドディレクトリにコピーされます。そこで再コンパイルされ、CGI アプリケーションにリンクされます。アプリケーションを .int ファイルまたは .gnt ファイルとしてパブリッシュしたい場合には、UNIX システム上で Acccgi ランタイムサポートモジュールを手動でリビルドする必要があります。これを行う場合は、アプリケーションとともに Acccgi モジュールを出荷する必要があります。

## 4.4 NetExpress IDE 設定と パブリッシャ

NetExpress IDE の設定には パブリッシャの操作に影響を与えるものがあります。

### 4.4.1 コンパイル指令

「プロジェクト、プロパティ、プロジェクト指令」ダイアログで設定されたコンパイル指令は、UNIX システムでコンパイル指令ファイルを作成するために パブリッシャで使用されます。指令ファイルは *projectname*.dir と呼ばれます。

NetExpress によりデフォルトで設定された次のコンパイル指令は、パブリッシュ中に無視されます。

- ENSUITE
- WB3
- FLAGEUC
- % 文字を含む指令。% は NetExpress の変数エントリを示すために使用されます。たとえば、%TARGETDIR です。

「ビルド設定、指令」ダイアログで指定されたコンパイラ指令はチェックフェーズ指令としてだけ、UNIX システムのコンパイラコマンド行にインクルードされます。これらの指令をこのダイアログから UNIX のコンパイルプロセスの生成フェーズに渡すことはできません。

### 4.4.2 エントリポイント

UNIX システムにパブリッシュされたシステム実行ファイルの場合には、エントリポイントは「プロジェクト、ビルド設定」ダイアログの [リンク] タブの [エントリ ポイント名] で定義されています。

## 4.5 環境変数

UNIX 上のリモートシェルサーバーは、標準的なシステム初期化ファイル (たとえば、.profile, .cshrc, .kshrcなど) を実行せず、最小限の環境変数の設定のみを提供します。 パブリッシュ中の環境に対して、変数の変更または追加指定を行うには、ファイル .mfenv に環境変数を記述します。 $HOME ディレクトリとビルドディレクトリの両方

で .mfenv ファイルを定義することができます。実際に、$HOME ディレクトリの .mfenv ファイルはグローバルファイルとして機能します。プロジェクトがパブリッシュされると、UNIX オプションは $HOME ディレクトリの .mfenv ファイルを検索します。ファイルが見つかった場合には、その環境変数はビルド環境に追加されます。その後、UNIX オプションはビルドディレクトリの .mfenv ファイルを検索します。.mfenvファイルが見つかった場合には、その環境変数がビルド環境に追加されます。

.mfenv ファイルは、UNIX ボーンシェルの部分集合であり、次のものが含まれます。

- 第 1 列がハッシュ (#) 文字であるコメント行
- 次の形式の環境変数

  *variable*=*value*

*value* が別の環境変数を示す場合には、次のようになります。

- 環境変数は括弧または中括弧で囲まれている場合のみ評価されます
- 代替パラメータはサポートされていません

次に例を示します。

| | |
|---|---|
| `MYPATH=$PATH` | `$PATH` は展開されません。環境変数 `MYPATH` はリテラル値 `$PATH` に設定されます |
| `MYPATH=$(PATH)` | 変数 `$PATH` を現在の環境で調べます。 それが見つかった場合には、展開され、`${PATH}` の値に置き換わります。見つからない場合には、空白の文字列と見なされます。 |
| `MYPATH=$(PATH:="/usr/mybin")` | サポートされていません。環境変数 `PATH:="/usr/mybin"` を見つけようとします。これは失敗し、空白の文字列で置き換えられます。 |

## 4.6 システムコピーファイル

COBOL の構成要素には、アプリケーションをビルドするときにシステムコピーファイルを呼び出すものがあります。NetExpress では、これらのすべてのシステムコピーファイルが 1 つの場所に格納されます(NetExpess¥Base¥Source)。COBOL for UNIX では、システムコピーファイルは多数のディレクトリに格納され、拡張子 .cpy または .CPY が付けられています。

システムコピーファイルは、アプリケーションが UNIX システムにパブリッシュされるときに UNIX サーバーにコピーされません。COBOL for UNIX は固有のシステムコピーファイルを使用する必要があるからです。パブリッシュ操作中に、システムコピーファイルがコピーされなかったことを伝えるメッセージが IDE のパブリッシュペインに表示されます。

パブリッシャは、UNIX システム上でこれらのシステムコピーファイルを動的に検索することができないため、それらがディレクトリ $COBDIR/cpylib にあると想定します。UNIX システム上の パブリッシャが作成した Makefile は、システムコピーファイルがこのディレクトリにあると想定します。

ほとんどのシステムコピーファイルは $COBDIR/cpylib にあり、問題を起こすことはありません。ただし、必要なシステムコピーファイルを検索するときに問題が発生したためにビルドが失敗した場合には、システムコピーファイルをディレクトリ $COBDIR/cpylib にコピーし、`make` コマンドを使用してアプリケーションをリビルドする必要があります。また、NetExpress で使用されているシステムコピーファイルの名前の大文字と小文字が COBOL for UNIX のそれと異なる場合には、ビルドが失敗する可能性があります。この場合には、ファイル名を変更してください。代わりに、「パブリッシャ セットアップ」ダイアログのファイルマッピング機能を使用することもできます。

## 4.7 AIX へのアプリケーションのパブリッシュ

この章を読む必要があるのは､次の場合だけです。

- Server Express より前の COBOL for UNIX バージョンの UNIX システムへパブリッシュする場合
- UNIX システムのオペレーティングシステムが AIX である場合
- 共有ライブラリ (DLLs) を使用して作業している場合

この場合は、作成しようとしている共有ライブラリに対するエクスポートファイルが必要です。

AIX の場合だけ、作成しようとしている共有ライブラリに対するエクスポートファイルが必要です。 このファイルは自動的には作成されません。エクスポートファイルは AIX リンカの標準的な部分であり、1 行あたり 1 つのエクスポートされたシンボルを含む必要があります。エクスポートファイルの名前は、ファイル拡張子 .exp をもつターゲットのベース名です。

たとえば、共有ライブラリ mylib.dll を作成するプロジェクトがある場合には、そのプロジェクトは prog1.obj と prog2.obj を含みます。`prog` と呼ばれる有効なエントリポイントが 1 つだけあります。

AIX にパブリッシュするには、ファイル mylib.exp が必要です。このファイルは、テキスト `prog` を含む 1 行のテキストファイルです。これをプロジェクトに追加する場合には、パブリッシュするたびに AIX に自動的に転送されます。代わりに、AIX システム上でファイルを維持するだけでもかまいません。

---

備考: エクスポートファイルは AIX の場合だけ必要です。他の UNIX システムにパブリッシュする場合には必要ありません。

---

## 4.8 状態メンテナンスルーチンの使い方

状態メンテナンスライブラリルーチン (MF_CLIENT_STATE_*n*) を使用すると、web サーバー上でユーザーとアプリケーションの状態情報を格納できるインターネットアプリケーションを作成することができます。アプリケーションでこれらのルーチンを使用した場合には、これらのルーチンのランタイムサポートモジュールがアプリケーションとともに UNIX システムにパブリッシュされていることを確認することが必要です。

これを行うには、ファイル sstate.int を ¥NetExpress¥Unix¥CGI-SUP からプロジェクトにコピーします。アプリケーションをパブリッシュするとき、sstate.int もまた UNIX システムにパブリッシュされます。

# 第5章　NetExpress への UNIX アプリケーションのインポート

UNIX システムに常駐している開発済みのアプリケーションを編集するために NetExpress を使用したい場合には、そのアプリケーションを NetExpress にインポートする必要があります。インポートは一度だけ行います。アプリケーションが PC 上にインポートされると、NetExpress を使用してそのアプリケーションを保守することができます。

ほとんどの複雑な UNIX アプリケーションは、複数のアプリケーションモジュールから構成されています。アプリケーションモジュールは、アプリケーションの明確に区別されたプログラムのグループと見なすことができます。たとえば、会計のアプリケーションは、購買伝票、元帳、販売伝票などのいくつかのモジュールから構成されています。各アプリケーションモジュールは多くのプログラムから構成されている場合があります。各モジュール (購買伝票など）には固有のディレクトリ構造があり、そこでモジュールが保守管理されています。一般的に、アプリケーションソースコード、コピーライブラリ、データファイルおよび実行可能ファイル用に別々のディレクトリがあります（中間コード、生成コード、実行可能コードについてもディレクトリが別になっていることがよくあります）。アプリケーションをインポートするときには、複数のソースディレクトリを指定して、複数のターゲットディレクトリへのコピーを指定することができます。

アプリケーションをインポートする前に、NetExpress の前提に注意してください。ソースコードが複数ディレクトリにある場合 (これは好ましい作業形態ではありません) でも、各プロジェクトは 1 つのディレクトリに存在します。 NetExpress はプロジェクトの全モジュールを同じディレクトリにビルドするので、アプリケーション内でのファイルを置く方法に制限があります。 たとえば、別のディレクトリに」同じソースファイル名をもつことはできません。 UNIX からアプリケーションをインポートしたいときは、最小限のソースディレクトリ数にするようにし、アプリケーション内のモジュールごとに別々の NetExpress プロジェクトを作成することを推奨します。

[Unix オプションのインポート] ウィザードを使用すると、UNIX アプリケーションを NetExpress にインポートすることができます。しかし、アプリケーションモジュールを開発したり保守したりするには、NetExpress を使用する前に、プロジェクトに手作業で変更を行う必要があります。

インポートウィザードを起動するには、[UNIX インポート] をクリックします。

---

備考: インポートウィザードを開始する前に、プロジェクトのソースコードすべてが、Samba 経由でネットワーク上で共有可能であることを確かめてください。PC からアクセス可能である必要があるからです。 Samba のインストールと構成の詳細については、付録「SCP と Samba のインストール」を参照してください。

---

インポートウィザードの手順は次のとおりです。

### 手順 1 - プロジェクトの詳細

NetExpress プロジェクトとプロジェクトディレクトリに名前をつけます。 NetExpress では、プロジェクトディレクトリは特に重要です。NetExpress は、プロジェクトディレクトリにソースコードをおき、アプリケーションをビルドすることを前提としています。

### 手順 2 - ソースディレクトリ

プロジェクト用のソースコードがある UNIX システム上のディレクトリすべてを、入力します。 コピーファイルディレクトリもすべて含む必要があります。

これらのディレクトリにファイルが存在していない場合に、手順 4 でファイルを追加することはできません。

### 手順 3 - ターゲットディレクトリ

入力した各ソースディレクトリに対して、ディレクトリ内のファイルをコピーする場所を指定することができます。デフォルトはプロジェクトディレクトリです。 ターゲットディレクトリの値どおりに、プロジェクトディレクトリの値を保持し、必要な場合だけ変更するようにしてください。

### 手順 4 - ソースファイルの指定

プロジェクトを作成するソースファイルすべてを入力します。 入力することができるのは、手順 2 で指定したソースディレクトリのいずれかに存在するファイルだけです。

[コピーファイルを走査] ボタンをクリックすると、インポートウィザードが入力したファイルすべてについて、COPY 文を含んでいるかどうか調べます。走査が終わると、参照するコピーファイルを位置指定しようとします。 コピーファイルの走査を行うのは、手順 2 で指定したソースディレクトリだけです。 みつかったコピーファイルは、ソースファイルリストに追加されます。 参照するコピーファイルがCOPY 文によってみつからなかった場合は、これらのファイルが表示されます。

---

備考: コピーファイルの走査は、参照するコピーファイルを文字どおり、位置指定するだけです。例: `COPY "filename.cpy"`

---

列 [ファイルタイプ] は、ファイルがテキストかバイナリかを指定します。デフォルトはテキストです。 インポート変換操作 (手順 5 で指定します) に対する有効なファイルは、テキストファイルです。 バイナリファイルは、変更されずにソースからターゲットにコピーされます。

### 手順 5 - インポート変換

手順 4 でタイプをテキストと定義されたソースファイルに対して、インポートウィザードで処理することができま

す。

ここで指定する探索/置換パターンは、*探索/置換パターンの設定*で説明したものと同じです。 正規表現についての詳細は、付録「正規表現」を参照してください。

テキストファイルの UNIX 形式の改行区切りを、PC 形式の改行区切りに変換するには、「UNIX テキストファイルを PC 形式に変換」チェックボックスをチェックします。

テキストファイルを、UNIX システム上で使用される日本語 EUC 文字セットコードから、PC 上で使用される日本語 SJIS 文字セットコードへ変換するには、「テキストファイルを EUC から SJIS に変換」をチェックします。

---

備考: SJIS 文字コードも使用する UNIX システムもあります。 日本語プロジェクトをインポートしている場合は、どの文字セットでプロジェクトがエンコードされているかを確認してください。

---

**手順** 6 - **その他のプロジェクト設定**

UNIX 上に指令ファイルがない場合は、自動的に UNIX 上から NetExpress プロジェクトにチェッカー設定が追加されます。cobol.dir および cobopt ファイルは両方ともサポートされています。

---

備考: cobopt ファイルからはチェッカーオプションだけがインポートされます。 他のオプションは無視されます。

---

インポートウィザードがプロジェクトにソースファイルを追加して、各 COBOL ファイルに対してデフォルトのターゲットを作成するようにするには、「デフォルトターゲットを作成」をチェックします。 これをチェックしない場合は、ソースファイルはプロジェクトにコピーされますが、ターゲットは作成されません。

**手順** 7 - **確認**

[完了] をクリックすると、指定されたソースファイルがコピー・変換され、プロジェクトが作成されます。

インポートウィザードは、自動的に INTLEVEL"2" および WARNINGS"2" 指令をユーザーのプロジェクト設定に追加します。 さらに、手順 5 で「テキストファイルを EUC から SJIS に変換」をチェックした場合は、FLAGEUC 指令がプロジェクト設定に追加されます。

## 5.1 COBOL **コピーファイル**

指定したソースファイルが PC 上で複数のソースディレクトリへインポートされる場合は、NetExpress がコピーファイルを位置指定できるように COBCPY 環境変数を設定する必要があります。 COBCPY 環境変数は、NetExpress

IDE の「プロジェクトのプロパティ」で設定することができます。

## 5.2 COBOL データファイル

NetExpress プロジェクトディレクトリ以外のディレクトリに COBOL データファイルがある場合には、実行中に外部ファイル名マッピングを使用してデータファイルを検索することができます。外部ファイル名マッピングは、テキストファイル (マッパーファイル) 内でのファイル名マッピングの解決を有効化することにより、COBOL プログラムで使用された割り当てファイル名を物理的なファイル名に変換するための柔軟な方法を提供します。後でファイルを編集することで、ファイル名マッピングを変更することができます。たとえば、マッパーファイルには次のエントリを含むことができます。

```
apfile c:¥tmp¥unixapp¥ap¥data¥apfile
arfile c:¥tmp¥unixapp¥ar¥data¥arfile
```

外部ファイルマッパーを使用するには、指令 ASSIGN"EXTERNAL" を使用してプロジェクトをコンパイルし、ランタイム調整可能パラメータ `environment_mapper` を TRUE に設定する必要があります。使用するすべてのプロジェクトにおいて、マッピングファイルはメインプロジェクトディレクトリになければなりません。

外部ファイル名マッピングの詳細については、ファイル処理 (FHpubb02.htm) ブックのファイル名の章を参照してください。

## 5.3 他のプロジェクトでのプログラムの実行

プロジェクトが現在のプロジェクトの外部のプログラムを呼び出す場合には、呼ばれるプログラムの場所を COBDIR 環境変数に追加してください。パスが絶対パス名の場合、プログラムへのパスを完全に削除し、COBDIR 環境変数にそのパスを追加することができます。より柔軟なアプローチとしては、環境変数を使ってパスを置き換え、その環境変数を設定してから、NetExpress を呼び出す方法があります。

# 第6章　データベースアプリケーションでの COBSQL の使い方

COBSQL は、リレーショナルデータベースのベンダーが提供する COBOL プリコンパイラで機能するように設計された統合プリプロセッサです。 COBSQL は、次の環境での使用を目的としています。

- Oracle Pro*COBOL バージョン 1.8

UNIX プラットフォームに展開するアプリケーションを作成していて、Oracle リレーショナルデータベースにアクセスする場合には、COBSQL を使用する必要があります。

## 6.1 展開可能なデータベースアプリケーションの開発

NetExpress で COBSQL を使用して、UNIX システムに展開可能なデータベースアプリケーションを作成するには、まずデータベースソフトウェアを設定します。

1. UNIX システムに UNIX データベースサーバーをインストールします。詳細については、データベースサーバーのインストール指示を参照してください。
2. PC と UNIX 展開システムの両方にクライアントソフトウェアをインストールします。詳細については、データベースサーバーのインストール指示を参照してください。 データベースサーバーとクライアントが通信できることを確認します。
3. データベースライブラリにリンクして、UNIX プラットフォームに新しい COBOL ランタイムシステムを準備します。Makefiles (またはシェルスクリプト) は通常、データベースプリコンパイラに付属しています。データベースサポートモジュールを組み込む COBOL ランタイムを再リンクするためにこれらの Makefile を使用する必要があります。この再リンクは、各プラットフォームの各データベースに対して一度だけ実行します。

通常は、これらの手順を一度だけ実行する必要があります。UNIX に COBOL システムの新しいリリースをインストールする場合には、あるいはデータベースクライアントを更新する場合には、データベースサポートモジュールで COBOL ランタイムシステムを再リンクする必要があります。

NetExpress で COBSQL とデータベースプリコンパイラを使用してデータベースアプリケーションを開発します。NetExpress で COBSQL を呼び出す方法については、データベースアクセス (DBpubb02.htm) ブックを参照してください。通常、データベースプリコンパイラには埋め込み SQL 文を使用して COBOL プログラムをコーディングする方法を示すサンプルプログラムが付属しています。NetExpress でアプリケーションをアニメートまたは実行し、そのアプリケーションの動作を確認した場合は、データベースアプリケーションを UNIX システムにコピーします。

1. ファイル cobsql.dir をプロジェクトのソースプールに追加します。このファイルは UNIX に必要なすべての COBSQL 指令を含みます。このファイルの追加は一度だけ行う必要があります。

2. [UNIX セットアップ] をクリックします。このプロジェクトに合わせて パブリッシャを構成します（パブリッシャの設定の詳細については、*パブリッシャの設定*を参照してください）。特に、[サーバー上でビルド] をチェックします。これは、パブリッシャが UNIX システムにプロジェクトをコピーするが、プロジェクトをリビルドしないことを指定します。

   プロジェクトをパブリッシュします。ファイルが UNIX システムにコピーされ、Makefile が作成されますが、アプリケーションはリビルドされません。

3. UNIX マシンの publish ディレクトリにある、Makefile ファイルを新しいファイルにコピーします。たとえば、CSQLMakefile という名前でコピーします。

4. CSQLMakefile を変更し、プログラムをコンパイルするために使用するコマンド行に`k`オプションと`C"p(cobsql)"`指令を含むようにします。

   ```
   Animtst1.int: Animtst1.pco
   ./mfenv.sh cob Animtst.pco
   ```

   上記のコマンド行は、次のようになります。

   ```
   Animtst1.int: Animtst1.pco


   ./mfenv.sh cob –k Animtst.pco –C"p(cobsql)"
   ```

5. この修正した CSQLMakefile で `make` を使用して、UNIX システム上でこのプログラムのデバッグ可能なバージョンを作成します。

   備考: 必要に応じて、CSQLMakefile を正しく設定した場合には、パブリッシャを使用してこの手順を自動化することができます。ビルドが行われる前に実行する UNIX コマンドを パブリッシャに指定することができます。

   1. [UNIX セットアップ サーバー 設定 Prebuild] をクリックします。

   2. 「PreBuild コマンド」フィールドにコマンド `make –f`*`makefilename`* を入力します。*makefilename* は編集した Makefile の名前です（この例では、CSQLMakefile)。

      クリックします。

   3. [サーバー] タブをクリックし、[サーバー上でビルド] をチェックします。

   この場合には、プロジェクトをパブリッシュするときに自動的に生成されたビルドは失敗しますが、makefile を使用して作成したビルド (この例では CSQLMakefile) は、アプリケーションを正常にリビルドします。

6. UNIX システムのアプリケーションをアニメートまたは実行します。

7. アプリケーションが期待通りに動作することを検証するために、NetExpress でアプリケーションを実行した結果と UNIX システムでアプリケーションを実行した結果を手動で比較します。

## 6.2 UNIX の考慮事項

UNIX システムに展開したいアプリケーションを NetExpress で開発している場合には、次の点に注意してください。

- UNIX では、cob コマンドが受け付ける通常の拡張子は .cob と .cbl です。デフォルトの拡張子をオーバーライドするには、ファイル名の前に -k オプションを使用します。cob を構成し、.pco と .eco 拡張子をデフォルトのファイル拡張子のリストに追加することができます。詳細については、マニュアルの cob に関する節を参照してください。

- cob コマンド行で COBSQL に指令を渡す場合には、等号 (=) が必要な指令の前には 2 つの等号を指定する必要があります。これは、cob が 1 つの等号を削除し、その後の値を括弧で囲むためです。等号が 2 つ指定されている場合には、cob は最初の等号を削除し、2 番目の等号を COBSQL に渡します。次に例を示します。

  ```
  COBSQLTYPE=ORACLE
  ```

  上の COBSQL 指令は次のように入力します。

  ```
  COBSQLTYPE==ORACLE
  ```

  これは cob コマンド行で指定された指令にだけ適用されます。cobsql.dir ファイルの指令には適用されません。

- データベースサポートをインクルードする通常の方法では、COBOL ランタイムを再リンクして新しいランタイムを作成します。この新しいランタイムは、データベースアプリケーションを実行するときに使用します。

---

備考: COBOL ランタイムを再リンクするには、「C」開発システムのバージョンがマシンで使用可能でなければなりません。

---

- COBOL ランタイムを再作成するための makefile は、データベースプリコンパイラに付属しています。この makefile には、データベースランタイムサポートライブラリとモジュール名の定義が含まれるため、新しいランタイムはデータベースルーチンの名前を実行時に解決することができます。

- 新しいランタイムを使用してアニメートするには、COBSW 環境変数をエクスポートする必要があります。この環境変数は +A に設定します。アプリケーションをアニメートするには、次のように入力します。

*new-run-time program-name*

ここで、*new-run-time*は 再リンクされたランタイムの名前です。*program-name* はアプリケーションの名前です。

- 次のメッセージは通知のためのもので、致命的なエラーではありません。

  `* CSQL-F-016: UNIX Exec error  Return Code is` *`return code`*

  このメッセージは、エラーが発生した結果として、プロジェクトリコンパイルが正しく実行されなかったことを示します。

- COBSQL がプリコンパイラを実行できない場合は、次の COBSQL エラーメッセージが表示され、COBSQL が終了します。

  ```
  * CSQL-F-021: Precompiler did not complete -- Terminating
  ```

### 6.2.1 コマンド行の例

次の例は、UNIX で動作する Oracle 用に COBSQL プログラムをコンパイルするために必要なコマンド行を示します。

```
cob -a -v -k sample.pco -C "p(cobsql) cstop cobsqltype==ORACLE"
```

### 6.2.2 エラーメッセージ

次の COBSQL エラーメッセージは UNIX システムでだけ表示されます。

CSQL-F-002: Unable to connect with Micro Focus CDI2

初期化処理の一部として、COBSQL は内部 Micro Focus モジュール CD12 に接続しようとします。 CDI2 には、コンパイルされるプログラムの変数についての CSI 情報をもつ COBSQL があります。CDI2 はツールボックスまたは Object COBOL 環境の必須部分であるため、通常の実行時には常に存在しています。このエラーはメモリに問題がある場合にだけ表示されます。ツールボックス環境または Object COBOL 開発環境メニューシステムを一度シャットダウンし、再起動することで問題は解決します。

CSQL-F-016: UNIX Exec error Return Code is *return-code*

UNIX では、これは致命的なエラーではありません。データベースコンパイラがソースファイルのエラーを見つけた場合には、ゼロ以外の戻りコードを生成する可能性があります。このため、COBSQL はこのメッセージを表示して処理を続行します。

# 第7章　ヒントとトラブルシューティング

本章では、以下について説明します。

- UNIX オプションの使用時に役に立つヒント。
  UNIX オプションの使用時に発生する問題と解決方法。

## 7.1 ヒント

### 7.1.1 CGI アプリケーションのパブリッシュ

CGI アプリケーションがファイルシステム上の複数の場所に存在することが、よくあります。 たとえば、CGI プログラムは cgi-bin ディレクトリに、静的 HTML ページは別のディレクトリに、フォームはさらに別のディレクトリにある必要があるというようにです。

この問題への簡単な対処として、論理ディレクトリ機能を使用する方法があります。 「論理ディレクトリ」ダイアログで対になる名前を定義します。例: `HTML Pages` と `HTML Forms`。 「ファイルマッピング構成」ダイアログを表示して、HTML ファイルに論理名を割り当てます。 ワイルドカードを指定したり、シフトクリックしたりすると、複数のファイルを選択できます。

[サーバー設定 ディレクトリ] タブ上で、HTML ページと HTML フォームを含む物理ディレクトリを正確に入力します。 ファイルはその後、自動的にそこへコピーされます。

CGI プログラム自身は、常にビルドディレクトリにビルドされます。 これを自動的に正しい場所に置くには、[サーバー設定 ポストビルド] タブで、postbuild コマンドを指定する必要があります。

### 7.1.2 ファイルの自動変更

UNIX オプションの検索 / 置換パターンは正規表現を使用しており、とても強力です。 しかし、正規表現を使いこなすには、ある程度の経験が必要です。 経験を積むために最適な方法は試してみること、そして正規表現を使用したことのある人のアドバイスを受けることです。

検索 / 置換機能を使い始めたばかりの場合は、まず、文字列検索を使用してみてください。文字列検索は、直接、テキストをテキストに置換します。

おそらく最も理解しづらいのは、繰り返し文字を処理する正規表現です。 繰り返しメタキャラクタ (*+?) はすべて、直前の正規表現に対して機能します。 メタキャラクタ自身には、文字そのものとしての意味はありません。 直前の文字とは、単に 1 文字だけです。ただし、かっこを使用してグループ化された正規表現は例外です。注意してください。 ファイル名形式のパターンを使用しなれている場合は、混乱するかもしれません。 付録「正規表現」を参照して、正規表現の使用例を読んで理解してください。

## 7.2 トラブルシューティング

### 7.2.1 パブリッシュできない

サーバーコントロールプログラム (SCP) は、アプリケーションのパブリッシュ先の UNIX システムにインストールする必要があります。NetExpress と Unix の間のインターフェースを提供するからです。SCP をインストールしない場合には、アプリケーションをパブリッシュすることはできません。 詳細については、付録「SCP と Samba のインストール」を参照してください。

サーバーが `scp` の実行を拒否するメッセージが表示された場合は、ユーザの .rhosts 設定をチェックします。 構成方法の詳細については、付録 「SCP と Samba のインストール」 を参照してください。

### 7.2.2 .dll ファイルを含むアプリケーションをパブリッシュできない

この問題は、NetExpress と COBOL for UNIX の間の COBOL の機能の違いによって発生します。 NetExpress 上では、プログラムは .dll ファイルディレクトリを呼び出すことができます。 COBOL for UNIX 上では、共有オブジェクトをアクセス可能にするには、共有オブジェクトを実行形式 (またはランタイムシステム) にリンクする必要があります。

---

備考: Server Express は、NetExpress と同じ方法で共有オブジェクトを直接、呼び出すことができます。

---

このようなアプリケーションを UNIX にパブリッシュするには、IDE をうまく操作して、実行形式モジュールと .dll ファイルの間に従属関係を作成する必要があります。アプリケーションをパブリッシュしたときに、この従属関係が見つけられ、共有オブジェクトが実行形式モジュールに正確にリンクするようになります。

次の手順では、同じ NetExpress プロジェクト内の .exe ファイルによって呼び出される .dll ファイルがあると、仮定しています。

1. ビルドペインで .dll ファイルを選択し、右クリックして [ビルド設定] を選択します。
    1. [リンク] タブを選択します。
    2. [カテゴリ] プルダウンメニューから [高度な指令...] を選択します。
    3. 「一時リンカーファイルの保持」オプションをチェックします。
    4. ダイアログボックスを閉じます。
2. ビルドペインで .dll ファイルを選択し、右クリックして [オブジェクトをリビルド] を選択します。 または、[すべてをリビルド] をクリックします。 これで、.dll ファイルに対応するターゲットディレクトリが

作成されます。

3. ソースペインで、右クリックして [ソースプールにファイルを追加] を選択します。 ターゲットディレクトリに作成された .lib ファイルを選択します。 プロジェクトディレクトリにファイルをコピーするかどうか、NetExpress が尋ねてくるので、[いいえ] を選択します。

4. .lib ファイルをビルドペインにドラッグして、実行形式モジュールの一部に含めます。

5. これで、NetExpress 内にプログラムをビルドし、実行できます。

6. 必要な場合は、パブリッシャをセットアップします。 これで、このプロジェクトを UNIX へパブリッシュすることができます。 そして、共有オブジェクトにリンクされた実行形式が作成されます。

### 7.2.3 ユーザー ID をビルドエリアで変更できない、または共同作業者と共有できない

ビルドエリアは、1 人のユーザーだけに使用される傾向があります。 これはビルドエリアのロックが原因です。

共有ビルドエリアの主な問題は、ある人が行ったソースプログラムの変更が、別の人が同じビルドエリアで行ったソースコードの変更と混在することです。 この結果、微妙なエラーのせいでコンパイルに失敗してしまいます。

実際に共同作業者とビルドエリアを共有する必要がある場合は、共通ユーザー ID を設定して、自分のユーザーID も共通ユーザー ID も使用できるようにすることを推奨します。 同期的にユーザーの変更を保持するために、ソースコード制御システムを使用することを、強く推奨します。 [サーバー設定] タブ上で [パブリッシュ時にソースを検証] オプションを有効にする場合は、UNIX Option の期待するバージョンとサーバー上のファイルを比較して、違いのあるファイルを パブリッシャが通知してきます。

ビルドエリアをあるユーザー ID から別のユーザー ID に変更するには、現在、ロックを所有しているユーザーが、「サーバーの排他制御」 ダイアログを使用してエリアのロックを解除する必要があります。 新しいユーザーがパブリッシュに成功する前に、おそらく、ビルドエリア内のファイルの所有権を、古いユーザーから新しいユーザーに手作業で変更する必要があると思われます。 UNIX システムによっては、これには root アクセス権が要求されます。

### 7.2.4 システムコピーファイルの問題

パブリッシュ操作中に、パブリッシャが UNIX システム上でシステムコピーファイルを見つけることができないためアプリケーションが正常にビルドされなかったことを示すエラーを受け取る可能性があります。このエラーが発生した場合には、システムコピーファイルをディレクトリ $COBDIR/cpylib にコピーする必要があります。場合によっては、UNIX のシステムコピーファイルの拡張子の大文字小文字を変更する必要があります（たとえば、コピーファイルの拡張子が .cpyの場合は、.CPY に変更することが必要になる場合があります)。

詳細については、「アプリケーションのパブリッシュ」の章の「システムコピーファイル」の節を参照してください。

### 7.2.5 CGI アプリケーションの問題

CGI アプリケーションが正常に機能しない場合には、次のチェックを行ってください。

- HTML 出力フォームは CGI プログラムと同じディレクトリに存在する必要があります。
- フォームのファイル名は Form Designer で大文字でハードコーディングされるため、ファイル名マッピングが正しく構成されていることを確認してください。 Acccgi モジュールは小文字の .htm 拡張子をもつファイルだけを検索することができます。大文字の .HTM または小文字の .html は検索できません。
- ファイルをすべてのユーザーが読み取ることができることを確認します。たとえば、ファイルのアクセス権を次のように変更します。`chmod +r *.htm`

また、インターネットアプリケーション (PIpubb03.htm) ブックの次の節をお読みください。

- UNIX システムに CGI アプリケーションを展開する準備の詳細については、『展開手順ガイド (UNIXサーバ)』 を参照してください。
- UNIX への CGI アプリケーションのパブリッシュの詳細については、UNIX へのアプリケーションのパブリッシュを参照してください。

# 付録A: SCP と Samba のインストール

SCP と Samba は、異なるプラットホーム上のさまざまなバージョンの UNIX オペレーティングシステムに対して用意されています。これらのプログラムは NetExpress CD のディレクトリ /unix/*os_name* に入っています。ここで、*os_name* はオペレーティングシステムのニーモニックです。たとえば、SCO OpenServer 5 と IBM AIX V4.1 の場合、プログラムは次のディレクトリに格納されています。

| | SCO OpenServer V5の場合 | IBM AIX V4.1の場合 |
|---|---|---|
| SCP | `/unix/sco5/scp` | `/unix/aix413/scp` |
| Samba | `/unix/sco5/samba.tar` | `/unix/aix413/samba.tar` |

NetExpress で提供されている SCP と Samba のバージョンの詳細については、NetExpress CD の /unix ディレクトリの readme.txt ファイルを参照してください。

---

備考: NetExpress CD から UNIX システムに SCP と Samba ファイルをコピーする方法としては、次の 2 通りの方法を推奨します。

1. NetExpress CD を UNIX システムに直接マウントし、標準的な UNIX コマンドを使用してファイルをコピーすることができます。NetExpress CD は、すべての UNIX システムでサポートされている a 形式で提供されています。一般的に、CD をマウントするコマンドは、

   ```
   mount /dev/cd0 /cdrom
   ```

   です。 オペレーティングシステムに必要な特定のコマンドについては、COBOL Developer Suite for UNIX または Object COBOL Developer Suite に添付されている*重要な CD-ROM情報*を参照してください。

2. Windows 95 と Windows NT の`ftp` コマンドを使用して、ネットワークを通して PC から UNIX にファイルをコピーすることができます。ファイルをコピーするときには、必ず`binary`フラグを設定してから転送を開始してください。

---

## A.1 SCP のインストール

SCP プログラムは、NetExpress UNIX オプションと UNIX オペレーティングシステムと COBOL 製品の間の標準的なインターフェースを提供します。

パブリッシャを正しく機能させるには、SCP をインストールする必要があります。

次の手順に従って、SCP をインストールします。

1. SCP プログラム (scp と呼ばれます) のバージョンを選択します。 使用しているオペレーティングシステムに適したプログラムを NetExpress CD から選択し、UNIX システム上の一時領域にコピーします。たとえば、/tmp です。

2. UNIX システムに root としてログインします。

3. /usr/local/bin ディレクトリが存在することを確認します。このディレクトリが存在しない場合には、作成します。

   ```
   mkdir /usr/local/bin
   ```

   誰もが読込み可能なようにします。

   ```
   chmod 755 /usr/local/bin
   ```

4. scp を一時領域から /usr/local/bin へコピーし、すべてのユーザが scp を実行できるようにします。例：

   ```
   cp /tmp/scp /usr/local/bin/scp ; chmod 755 /usr/local/bin/scp
   ```

## A.2 SCP の構成

SCP プログラムは、NetExpress UNIX オプションによって UNIX リモートシェル (RSH) プロトコルを使用して実行されます。 ほとんどすべての UNIX システムは、デフォルトで RSH サーバープログラムを使用可能です。他の特別なサーバーソフトウェアをインストールしたり、構成したりする必要はありません。 しかし、UNIX オプションを使用するには、RSH セキュリティ構成がユーザーに PC から SCP プログラムを実行可能にしていることを確認する必要があります。

### クイックスタート

アプリケーションのパブリッシュ先の UNIX システムのユーザー ID のホームディレクトリに、.rhosts というファイルがあることを確認してください。 このファイルには、パブリッシュ元の PC の正式名を含んでいる必要があります。

### A.2.1 RSH セキュリティ機構

RSH セキュリティ機構は、UNIX システムの構成ファイルに基づいて「ユーザー等価性」を確立することによって働きます。 ユーザーが等価である場合は、UNIX システムは呼び出されたプログラムへのアクセス権を、パスワードを要求ずに付与します。 .rhosts と hosts.equiv ファイルは、この等価性を制御するために使用されます。これらの 2 つのファイルを *rhosts ファイル* と呼びます。

rhosts ファイルは、オリジナルのバークレイ UNIX 版に由来しますが、すべての UNIX バージョンに広まりました。その過程で複数の異なるバージョンが生じています。SUN 拡張は NIS をサポートしており (旧名 Yellow Pages)、よく知られています。しかし、これらもまた、UNIX のさまざまな異なるバージョンを広めました。

UNIX オプションが UNIX システムに接続するとき、「サーバー設定」ダイアログで入力したユーザー ID を提供します。 UNIX システムは、ユーザの PC の正式名を IP アドレスに基づいて決定します。これには、逆調査という技術を使用します。 それから、ユーザーの正式マシン名とユーザー ID を使用して、アクセスを許可するかどうかを次の方法で決定します。

1. /etc/hosts.equiv があるかどうかをチェックします。 ファイルが存在する場合は、次の形式のテキストファイルである必要があります。

   *Machine-Name* [*User-ID*] [*#Comments*]

2. マシン名の次にユーザー ID が指定されていない場合は、全ユーザーが有効であるとみなされ、アクセスを許可されます。

3. PC マシン名とユーザー ID が hosts.equiv ファイルの行のどれかに一致する場合は、アクセスを許可されます。

4. /etc/hosts.equiv 内のマシン名とユーザー ID が、 ユーザーのPC マシン名またはユーザー ID に一致しない場合は、サーバーはリクエストで指定されたユーザー ID の HOME ディレクトリ内に .rhosts というファイルがあるかどうかチェックします。 .rhosts ファイルは、hosts.equiv ファイルと同じ形式です。

   ---

   警告: .rhosts ファイルは、ユーザー ID に所有され、ユーザー ID が所有するディレクトリに存在する必要があります。また、グループまたは全ユーザが書き込み可能であってはいけません (つまり、パーミッションは `rw-r--r--` である必要があります)。 また、シンボリックリンクであってもいけません。

   ---

5. PC マシン名が .rhosts ファイルのいずれかの行に一致した場合は、アクセスを許可されます。

6. 上記のどの手順でもアクセスを許可されなかった場合は、この時点でアクセスを拒否されます。

.rhosts ファイルのユーザー名フィールドは、UNIX ユーザーのために設計されています。あるシステムにログインするユーザーがリモートシェルを使用したり、別のユーザーとして他のシステムにリモートコピーを実行したりするかもしれないからです。UNIX オプションでは、.rhosts ファイルユーザー名フィールドは必要ありません。 アクセスしようとしている HOME ディレクトリのユーザー ID をいつも提供するからです。

RSH に対して主な SUN 拡張は、rhosts ファイルの有効マシンリストにシンボル (+) を追加しています。 これは、NIS 設定で使用するために設計されており、「すべての有効なマシン」を意味します。しかし、非 NIS 設定では「すべてのマシン」を意味します。 一般に、システムのセキュリティを混乱させるので、このシンボルの使用は避けるべきです。

ほとんどの場合、UNIX オプションに対する rhosts ファイルは次のように設定します。

- hosts.equiv ファイルは、一般にシステム管理者がリモートシステム上の全ユーザーが等価であることを宣

言するために変更されているだけです。 これは PC クライアントシステムでは推奨できません。PC 上ではどのユーザーも指定することができ、UNIX システムはそれを受け入れるからです。 悪意の PC ユーザーがシステム上の全ユーザーアカウント (root を除く) にアクセスすることを、無制限に許してしまいます。

- ユーザー ID 用の $HOME/.rhosts ファイルは、パブリッシュに使用中の PC のマシン名を含む必要があります。 ユーザー ID は必要ありません。

  サーバーが SUN 拡張をサポートしている場合は、.rhosts ファイルに + を追加して、ユーザー ID に対するマシン名のチェックを不可能にしたいことがあります (rhosts のマンページをチェックして + をサポートしているか調べてください)。

### A.2.2 .rhosts ファイル用の正式マシン名の決定

.rhosts ファイルに入力する PC のマシン名は、UNIX によって決定された正式なマシン名である必要があります。PC の通称は問題ではありません。名前は、IP アドレス接続に基づいて UNIX サーバーによって決定されます。 ほとんどのインストールでは、サーバーが名前を決定し、クライアント名は同じである必要があります。

システムの正式名を決定するには、まず、PC の IP アドレスを決定します。

---

備考: Windows 95 と Windows NT の構成ダイアログは異なります。 また、構成ダイアログはさまざまなサービスパックで更新されています。 したがって、使用する必要のあるダイアログは微妙に異なります。 以下の手順は、Windows NT V4.0 サービスパック 3 とインターネットエクスプローラ V4.01 をインストールして使用して作成しました。

---

1. [スタート] ボタンをクリックし、[設定] を選択します。
2. [コントロールパネル] をクリックし、[ネットワーク] をクリックします。

   ---

   備考: 識別 タブの コンピュータ名 フィールドは、TCP/IP 名とは関係のない NetBIOS 名です。

   ---

3. Windows NT では、プロトコル タブをクリックし、TCP/IP プロトコル をクリックし、プロパティ ボタンをクリックします。

   Windows 95 では、インストールされたネットワークコンポーネントがリストボックスに表示されるので、TCP/IP プロトコルを選択して、[プロパティ] ボタンをクリックします。
4. [IP アドレスを自動的に取得] ボタンがチェックされている場合は、動的に IP アドレスが割り当てられま

す。次の節 *「動的 IP アドレスの割り当て」*へ進んでください。

5. 「IP アドレスを指定」 がチェックされている場合は表示された IP アドレスの値を書き留めて、次へ進みます。

まず、UNIX マシンにログインします。 UNIX システムがマシン名と IP アドレスをクロスリファレンスするために使用している方法は、主に 3 つあります。

1. hosts ファイル。IP アドレスとマシン名を各行に含む単純なテキストファイルです。 通常、/etc/hosts にあります。

2. DNS (Domain Name System)。 インターネットによって使用されるシステムです。 世界中にある特別なサーバーを構成し、名称とアドレスを解読してお互いに接続します。 最も普及した方法となりつつあります。

3. NIS (Network Information System)。 NIS (旧名 Yellow Pages) は、ホストファイル、パスワード、グループ、エイリアス、サービス、その他の分散データベースです。

正式なホスト名を決定するには、UNIX システムで使用する名前の解決方法を決定する必要があります。そして、その方法を使用して、PC の IP アドレスに基づいた名前を調査します。

システム管理者に質問することもできます。システム管理者は、.rhosts ファイルに何を入力すべきか教えてくれます。

#### NIS が 構成されたかどうかを確認する方法

/etc/nsswitch.conf というファイルがあるかどうか確認します。 存在した場合は、`hosts:` で開始する行を探してください。 たとえば、次のような行です。

```
hosts: xfn nisplus dns [NOTFOUND=return] files
 hosts: xfn nis [NOTFOUND=return] files
 hosts: files
```

この行は、NIS、DNS および /etc/hosts 内のファイルを使用して、ホスト名が解決される順序を決定しています。

ユーザの正式名を決定するには、nsswitch.conf で定義された名前解決の順序に従います。

#### DNS が構成されているかどうかをチェックする方法

/etc/resolv.conf というファイルがあるかどうかチェックします。 存在する場合は、DNS は構成されています。 このファイルの内容はここでは重要ではありません。

### A.2.2.1 NIS を使用した正式名の決定

ypcat コマンドとともに NIS を使用して、正式名を決定することができます。 たとえば、PC の IP アドレスが `204.160.128.10` の場合は、次のように入力します。

```
ypcat hosts | grep 204.160.128.10
```

IP アドレスに関連する名前が複数ある場合は、最初の名前が正式名です。

#### A.2.2.2 DNS を使用した正式名の決定

nslookup コマンドとともに DNS を使用して、正式名を決定することができます。 このコマンドは、DNS サーバーに問い合わせを行う一般的な方法です。 たとえば、PC の IP アドレスが `204.160.128.10` である場合、次のように入力します。

```
nslookup 204.160.128.10
```

入力した IP アドレスの名前とアドレスの次に、情報を取得した DNS サーバーの IP アドレスと名前が表示されます。 返却される名前は、いつも正式名です。

#### A.2.2.3 /etc/hosts を使用した正式名の解決

hosts ファイルは単純なテキストファイルなので、直接、調べることができます。 たとえば、次のように入力します。

```
grep 204.160.128.10 /etc/hosts
```

IP アドレスに関連する名前が複数ある場合は、最初の名前が正式名です。

正式名を解決できない場合は、PC に正式名がない可能性があります。 この場合は、(ドット区切りの 10 進で) IP アドレスを直接 .rhosts ファイルに追加することができます。 しかし、この方法はマシンを特定します。 システム管理者に依頼して、会社のマスターマシン名テーブルにユーザーの PC 用の正式名を追加してもらう方がよいでしょう。

### A.2.3 IP アドレスの動的割当て

.rhosts アクセス機構は、IP アドレスの動的割当てをサポートしていません。 完全なセキュリティ機構があり、マシン名 (および IP アドレス) を定数で定義していると仮定します。

ユーザのネットワークシステムが DHCP (または BOOTP のような他の動的 IP スキーマ) を使用している場合は、静的 IP アドレス割当てが可能かどうかをネットワーク管理者に確認してください。 可能な場合は、静的 IP アドレスを取得し、正常な方法で .rhosts を構成してください。

---

備考: シリアルライン経由でダイアルアップしている場合は、おそらく PPP または SLIP を使用しています。その場合は、まず確実に IP アドレスの動的割当てをしています。

---

動的 IP アドレスを用いて UNIX オプションで作業するには、rhosts ファイル内でリストされている現在のマシン名を知る必要があります。 これには方法がいくつかありますが、利便性とセキュリティのどちらをとるかによります。

- オプション 1: マシン名チェックを使用不能にする

  セキュリティを考慮せず、サーバーが rhosts ファイルに対して SUN の + 拡張をサポートしている場合は、+ を .rhosts ファイルに追加し、そのユーザー ID を使用してパブリッシュしようとしているホストすべてが成功するようにします。 安全性は低いけれども、簡単な方法です。

- オプション 2: 動的 IP アドレスすべてにユーザーディレクトリへのアクセスを許可する

  .rhosts ファイルへのサブネット上に存在する動的に割り当てられた IP アドレスすべてに対して、マシン名を追加することができます。 たとえば、ネットワークが 動的 IP マッピング用に *x.x.x.*100 から *x.x.x.*120 を割り当てられ、dhcp100 から dhcp120 を命名したとします。

  dhcp100 から dhcp120 の名前をすべて .rhosts ファイル に追加した場合は、これらのアドレスのどれを割り当てられてもパブリッシュすることができます。 同じサブネット上で動的に IP アドレスを割り当てられた人は、ユーザとまったく同じパーミッションを持ちます。

  おそらく、ローカルな IT 部門に質問して、動的に割り当てられるアドレスを決定する必要があります。

- オプション 3: 動的に .rhosts ファイルを変更する

  最も安全性の高い方法です。しかし、最も不便な方法でもあります。 PPP を使用してダイアルアップしたり、PC をリブート (DHCP) したりするときに必ず、新規に IP アドレスを割り当てることができます。 したがって、リブートしたりダイアルアップしたりするたびに、ユーザは .rhosts ファイルを編集して、古いマシン名を削除し、新しく書き換える必要があります。

---

備考: 実際は、DHCP はリブートしなくても IP アドレスを変更できます。 しかし、ほとんどの環境ではこの方法をとることは、まずありません。

---

このオプションを使用するには、次の手順で実行します。

1. 端末エミュレータを使用して、UNIX システムに Telnet で接続します。
2. 現在の Windows IP アドレスを決定します。
   - Windows 95 上では、¥windows ディレクトリにある winipcfg.exe プログラム (Microsoft 提供) を使用します。

- Windows NT 上では、¥winnnt¥system32 にある ipconfig.exe プログラムを使用する必要があります。

3. UNIX システム上では、現在の実際の Windows IP アドレスであると UNIX システムがみなすマシン名を決定する必要があります (詳細については、節*「.rhosts ファイル用の正式マシン名の決定」*を参照)。

    - ユーザのシステムが、DNS で構成されている場合は (ファイル /etc/resolv.conf が存在する)、nslookup コマンドを使用して名前を決定することができます。 `nslookup` の後に、ユーザの現在の Windows IP アドレス (ドットで区切った 10 進で) を入力します。すると、IP アドレスに対応するマシン名が表示されます。

    - hosts ファイルを使用している場合は、hosts ファイルを編集したり、`grep` を使用して Windows IP アドレスをみつける必要があります。

4. .rhosts ファイルを編集して、古い動的 IP アドレス名を削除します。 新しい名前を追加してファイルを保存します。

5. これでパブリッシュすることができます。

## A.3 Samba のインストール

Samba は、標準的な PC 形式のネットワークを使用して、UNIX ファイルやプリンタを PC と共有できるようにします。 Samba は、UNIX マシンから PC へのアプリケーションのインポートを要求します。

---

備考: Samba は、いわゆる「TCP/IP 上の NetBIOS」を実行します。これは、UNIX 用の NetBIOS サポートの実行方法の 1 つです。 システムに他の実行方法をパッケージしているシステムベンダーもあります。 複数の実行方法を可能にしようとする場合は、2 つめの方法は初期化に失敗します。このサービス用のネットワークポートが使用中だからです。

既存の TCP/IP 上の NetBIOS パッケージがある場合は (ユーザ自身の Samba バージョンも含む)、UNIX オプションのインポート機能は既存のものを使用します。UNIX オプションは、API への標準的なファイルアクセス経由で UNIX ファイルをコピーする必要があるだけだからです。

Micro Focus は、NetExpress CD に付属する Samba 製品以外のパッケージを使用して発生した問題については、サポートしません。

---

Windows のネットワークに精通していない場合は、Windows のエクスプローラを使用してネットワークドライブをマップする方法や、net use コマンドの詳細については、Windows のマニュアルを参照してください。

UNIX システムに Samba をイントールする方法を次に示します。

1. 使用しているオペレーティングシステムに適した Samba のバージョンを NetExpress CD から UNIX システムの一時ディレクトリにコピーします。

2. Samba の構成ファイルがない場合は、構成ファイルのサンプルを含む tar ファイルを ¥unix¥samba¥smbconf.tar から、UNIX システム上の一時領域へコピーします。

3. ルート権限があることを確認します。

4. すでに Samba が実行されていないことを確認します。`ps` コマンドを使用して、実行中のプロセスを調べます。たとえば、コマンド `ps -eaf | grep mbd` を入力して、`nmbd` または `smbd` というプロセスが実行されていないことを確認します。`ps` コマンドでこれらのどちらかのプロセスが表示された場合には、システムにすでにインストールされている Samba が実行されているため、インストールを中止する必要があります。

   現在インストールされている Samba を更新したい場合には、更新するときに現在のユーザーの作業を中断しないようにする必要があります。再び必要なる場合のために、既存の Samba ファイルを安全な場所にコピーします。

   `kill` コマンドを使用して既存の `nmbd` と `smbd` プロセスを終了してから、この後の指示に従ってインストールを実行します。マスターデーモンプロセスだけを kill する必要があります。つまり、親プロセス ID (PPID) が 1 (init プロセス) であるプロセスです。これらの init プロセスは他のすべての Samba プロセスをシャットダウンします。

5. コマンド `tar xvf samba.tar` を使用して、tar アーカイブから Samba ファイルを抽出します。

6. `installsmb.sh` を実行してバイナリを /usr/local/samba に、マニュアルページを /usr/local/man にそれぞれインストールします。

7. 有効な Samba 構成ファイルがない場合には、サンプルファイルを tar ファイルから展開します。

   ```
   tar xvf smbconf.tar
   ```

   一連の smbconf.*description* ファイルが作成されます。 各構成ファイルの先頭には、その目的に応じてコメントがあります。できるだけ早く開始したい場合は、smbconf.basic を選択し、/usr/local/samba/lib/smb.conf へコピーします。

   smb.conf ファイルの編集方法についての詳細は、次の節の*「Samba の構成」*を参照してください。

8. ネームデーモンを開始します。`/usr/local/samba/bin/nmbd -D`

9. セッションデーモンを開始します。`/usr/local/samba/bin/smbd -D`

10. エラーメッセージについては、/usr/local/samba/var のログファイル log.smb と log.nmbをチェックしてくだ

さい。

11. smbclient コマンドを使用して、定義した共有ファイルが動作していることをチェックしてください。 マシン名が unixserv である場合は、次のように入力します。

    ```
    /usr/local/samba/bin/smbclient -L unixserv
    ```

    パスワードを求められた場合は、Enter キーを押します。 エラーが生じた場合は、Samba のログファイルをチェックしてください。

12. Samba が開始して実行中であることを確認したら、ユーザのシステムのスタートアップファイルに Samba を開始するコマンドを追加する必要があります。 手順については、システム管理者または UNIX システムのマニュアルを参照してください。 一般にシステムのスタートアップファイルの場所は、/etc/rc2.d です。

## A.4 Samba の構成

Samba は複合製品であり、Samba 管理者が変更可能な 100 以上の異なる構成のオプションを持っています。 その目的は、異なる構成すべてをカバーすることではなく、Samba サーバーのセットアップ時に生じる共通の問題を解決することです。

Samba のマンページは Samba バイナリと同時にインストールされ、詳細な参照情報を含んでいます。 マンページにアクセスするには、man コマンドがページを見つけられるように MANPATH 環境変数に /usr/local/man を追加します。 最も重要な man ページは次のとおりです。

- samba - Samba の概要
- smbd - セッションマネージャデーモン
- nmbd - ネームマネージャデーモン
- smb.conf - Samba の構成ファイル

さらに、Samba にはソースコードの一部としてたくさんのマニュアルと FAQ があります。 Samba のソースコードは NetExpress CD の ¥unix¥samba¥sambasrc.tar に付属しています。 ソースコードを展開すると、マニュアルは docs サブディレクトリにあります。マニュアルは、初歩的な問題から高度に技術的な問題まで説明しています。Samba について問題がある場合、または構成方法についてもっと詳細に調べたい場合に、非常に役立ちます。 このマニュアルを参照しても Samba の構成に問題がある場合は、docs ディレクトリに DIAGNOSIS.txt というとても役に立つマニュアルがあります。

### A.4.1 ゲストアカウント

Samba のセットアップ時に生じる共通の問題の 1 つに、ゲストアカウントの値があります。ゲストアカウントは smb.conf ファイルで指定されます。 このユーザー ID はクライアントが利用できる共有リストのようなものとして、Samba が使用します。これは、一般公開された共有ファイルを実際のゲストユーザーとして明確に使用する場合と

同じようなものです。

smb.conf ファイルで指定された値が存在し、有効な UID をもっていることは非常に重要です。ほとんどの UNIX システムは、特権のないネットワークユーザーをデフォルトでもっています。 一般例として、nobody、nouser および network がいろいろな UNIX システムで提供されています。 システムの /etc/passwd ファイルを確認して、どれがサポートされているか、また、smb.conf ファイルが正しい値を持っているかを確認してください。 適切なユーザー名を持っていない場合は、作成する必要があります。

---

備考: HP-UX では、nobody アカウントの UID (時には GID も) は、負の値です (これは違法です)。 Samba へのゲストアカウントにこのユーザー名を使用したい場合は、nobody アカウントの UID と GID を未使用の正の数値に変更する必要があります。

---

## A.4.2 ユーザー認証

PC が OSR2 より前のバージョンの Windows 95、または Windows NT V4.0 SP3 を使用している場合は、Samba でのユーザー認証に問題は生じません。

しかし、Windows 95 OSR2 および Windows NT V4.0 SP3 のリリースにおいて、Microsoft は、クライアントシステムが NetBIOS サーバーにユーザー認証される方法を変更しました。 変更点は、サーバーに送信されるパスワード情報が、非暗号化形式から暗号化形式に変わったことです。Microsoft が使用している暗号化パスワード形式は、UNIX passwd ファイルが使用しているものと互換性があります。 2 つの暗号化パスワード文字列が一致するかどうかは不明です。

---

備考: 非暗号化パスワードの使用は、UNIX の telnet または ftp セッションでパスワードを入力するのと同様に安全です。

---

Windows 95 と Windows NT の上記リリースに対して、Samba セキュリティを構成する方法は 3 つあります。

- Windows クライアントが非暗号化パスワードを再度、使用できるようにする
- UNIX 上に smbpasswd ファイルを構成する
- Windows NT ドメインを使用して認証する

詳細については、次の節を参照してください。

### A.4.2.1 Windows クライアントが非暗号化パスワードを再度、使用できるようにする

Windows クライアントの古い非暗号化パスワードサポートをレジストリ設定経由で、再び使用可能にできます。 Windows 95 と Windows NT のレジストリ設定は、docs ディレクトリ内の Samba ソースコードに付属しています。

**利点**:

- UNIX passwd データベースが、ユーザーを認証するために使用されます。 ユーザーの UNIX パスワードと Samba パスワードはいつも同じです。

**欠点**:

- NT サーバーの新バージョンは、このレジストリ設定から非暗号化パスワードを受け付けず、ログインを拒否します。
- レジストリの変更は、各クライアントシステム上で適用する必要があります。

Samba **の設定**:

```
[global]
 security=user
 encrypt passwords = no
```

### A.4.2.2 UNIX での smbpasswd ファイルの構成

Samba は、NetBIOS 形式の暗号化パスワードをサポートしてビルドされています。 NetBIOS 形式のパスワードは、特別な smbpasswd ファイルに格納されます。 smbpasswd ファイルは、smbpasswd を使用して手作業で作成されます。

**利点**:

- Windows クライアントを変更する必要がありません。

**欠点**:

- UNIX パスワードと NetBIOS パスワードは異なるファイルに保持されるので、管理上の付加が増大し、パスワードが違ってしまう可能性があります。

Samba **の設定**:

```
[global]
 security=user
 encrypt passwords = yes
```

**方法**:

- root パーミッションをもっていることを確認してください。
- smbpasswd 用の専用ディレクトリを作成します。 デフォルトでは、`/usr/local/samba/private` です。 このディレクトリは root によって所有され、root だけがアクセス可能である必要があります。 パーミッションは、`rwx------` です。
- smbpasswd コマンドを使用して、要求されるユーザー ID を追加します。例 : `smbpasswd -a myuserid`

---

備考: Samba のソースコード docs ディレクトリには、役に立つヒントやプログラムがあります。ユーザー ID を既存の passwd データベースから smbpasswd へ移植するときに参考になります。さらに、クライアントからユーザーがパスワードを変更する方法についての詳細な説明もあります。

---

### A.4.2.3 Windows NT ドメインサーバーを使用した認証

Samba は Windows NT ドメインサーバーに対してユーザーを認証できます。 Samba は実際にはドメインに参加できませんが、確認のために暗号化パスワードを Windows NT サーバーに転送できます。

**利点**:

- 共通ネットワークパスワードが、Windows NT と UNIX ネットワークとで共有されます。
- Windows クライアントの変更は不要です。

**欠点**:

- Windows NT サーバーが必要です。

Samba **の設定**:

```
[global]
  encrypt passwords = yes
  security = server
  password server = nt-server-name
```

特定のユーザー ID とパスワードについて Windows NT の認証に失敗した場合は、セキュリティモードが `security = user`" に戻り、Samba は smbpasswd ファイルが存在する場合はこのファイルでユーザーを認証しようとします。

---

備考: セキュリティ認証方法の使用に関係なく、実際の UNIX ユーザー ID は認証されたユーザー名用のファイル /etc/passwd に存在する必要があります。 Samba がこのファイルを要求するのは、ユーザーに適切なパーミッションを設定したり、ファイル所有権のパーミッションを確立したりするためです。

---

## A.4.3 複数の IP サブネット

NetBIOS は、本来、NetBEUI 転送を使用する単一の孤立した LAN 上で、小規模なワークグループ用に設計されました。 この設計の結果として、NetBIOS は、マシン名と共有可能情報のリストを作成して、LAN 内の全マシンにブロードキャストします。

TCP/IP 上の NetBIOS の出現によって、ブロードキャスト方式は機能しなくなりました。 TCP/IP は、サブネット内でのみブロードキャストを行うプロトコルです。 他の方法は、異なる IP サブネット間では、マシン名と共有情報を配布する必要があります。 解決方法は、Windows Internet Naming Service (WINS) です。

複合的な IP サブネットがあるときは、サブネット間で情報を関連させるには WINS サーバーを持つ必要があります。 Windows NT と Samba は両方とも WINS サーバーを含みます。 一般に、Windows NT サーバーがある場合は、付属の Samba よりはむしろ WINS サーバーを使用してください。 WINS サーバーはネットワークごとに 1 つだけです (ただし、Windows NT は、バックアップ用 WINS サーバーを持てます)。

**クライアントの構成**:

どのクライアントシステムも TCP/IP プロパティで構成された WINS サーバーを持つ必要があります。 手作業で行うか、DHCP サーバー経由で中央設定されるかします。 クライアントシステムは、このサーバーに自身を登録し、マシン名と共有情報を探すときはこのサーバーに問い合わせます。

**Samba の構成**:

Samba の構成は、Samba と Windows NT (または リモートマシン上の Samba) のどちらを WINS サーバー上で実行しているかによります。

- Samba が WINS サーバーの場合:

```
[global]
  wins support = yes
```

- 他のマシンまたはリモートの Samba が WINS サーバーの場合:

```
[global]
  wins support = no
  wins server = wins-server-name
```

ここで *wins-server-name* は、WINS サーバーマシンの名前です。

### A.4.4 Samba のインストールの変更

基本的なインストール先 /usr/local/samba は、実際には Samba バイナリにコンパイルされています。 Samba のインストール先を変更したい場合は、付属の Samba ソースコードを再コンパイルするか、またはコマンド行と構成ファイルオプションでファイルの場所を変更することができます。

ほとんどの場合、簡単で信頼性が高いのは、新しい基本ディレクトリにして Samba ソースコードを再コンパイルする方法です。Samba Makefile を 1 行変更するだけです。

---

備考: Samba の機能によっては (例: 複合コードページのサポート)、機能させるためには、新しい基本ディレクトリで Samba を再コンパイルする必要があるものもあります。

---

Samba の再コンパイルが実用的でない場合は、次の例のように変更を加える必要があります。 この例では、新しいインストール先を /home/samba と仮定しています。

まず、nmbd および smbd コマンドに、オプションを追加して、新しい構成ファイルの場所と出力をログする場所を指定します。

```
nmbd -s /home/samba/lib/smb.conf -l
/home/samba/var/nmb.log -D
smbd -s /home/samba/lib/smb.conf -l
/home/samba/var/smb.log -D
```

次に、Samba 構成ファイル内で、[global] セクションに以下を追加します。

```
lock directory = /home/samba/var/locks
smb passwd file = /home/samba/private/smbpasswd
smbrun = /home/samba/bin/smbrun
```

# 付録B: 正規表現

UNIX オプションでは､インポート中またはパブリッシュ中に正規表現を使用して検索操作や置換操作を実行します。

検索操作や置換操作では、テキスト ファイルは、連続した行として読み取られます。各行は、適用可能なすべての検索パターンまたは置換パターンを使用して処理されます。これらのパターンが適用される順序は、UNIX オプションのセットアップ時に指定した順序によって制御されます。

---

注記: 各行は個別に処理されるので、複数行にわたって検索する可能性のあるパターンは指定できません。

---

正規表現の構文は、UNIX の `grep` コマンドの構文とよく似ています。正規表現には、通常の文字とメタ文字の両方を使用します。メタ文字を使用すると、他の通常の文字に関する特別な情報を伝えたり、その意味を変更したりすることができます。たとえば、PC で DOS プロンプトを使用したことのあるユーザーにはなじみ深い dir *.* コマンドのメタ文字は、アスタリスク (ワイルドカード) で、0 個以上の通常の文字を表します。

## B.1 検索パターン

検索パターンの定義に使用できるメタ文字は、次のとおりです。

| メタ文字 | 説明 |
|---|---|
| ^ | 行の先頭を検索します。文字クラスでクラスを無効化します。 |
| $ | 行末を示します。 |
| . | すべての文字を検索します。 |
| [ | 文字クラスの先頭を示します。 |
| ] | 文字クラスの末尾を示します。 |
| * | 直前にある正規表現を 0 個以上含む文字列を検索します。 |
| + | 直前にある正規表現を 1 つ以上含む文字列を検索します。 |
| ? | 直前にある正規表現を含まない文字列、または、1 つ含む文字列を検索します。 |
| \| | 左右にある正規表現を検索します。 |
| ( | サブ文字列の先頭を示します。 |

| | |
|---|---|
| ) | サブ文字列の末尾を示します。 |
| " | リテラル文字列の文字を区切ります。 |
| ¥ | エスケープ文字です。 |

## B.2 置換パターン

置換パターンの定義に使用できるメタ文字は、次のとおりです。

| メタ文字 | 説明 |
|---|---|
| & | 検索パターンと一致する文字列を示します。この後に 1 から 9 までの数字 (*n*) が続く場合、この文字列は、サブ文字列の数字 *n* と一致します。 |
| ¥ | エスケープ文字です。 |

## B.3 エスケープ文字

エスケープ文字は、メタ文字が持つ特殊な情報を無効化するために使用します。

たとえば、検索パターン $HOME には、特殊な意味を持つドル記号 ($) が使用されているため、これを検索することはできません。この文字列を正しく検索するには、¥$HOME のように指定します。バックスラッシュ (¥) は、これに続く文字の特殊な意味を無効化する文字です。

さらに、エスケープ文字を使用して、別の方法で表示しにくい、または、表示できない特殊文字を定義することができます。これらをエスケープ シーケンスと呼びます。使用できるエスケープ シーケンスは、次のとおりです。

| エスケープ シーケンス | 説明 |
|---|---|
| ¥b | バックスペース |
| ¥e | ASCII エスケープ文字 |
| ¥f | 用紙送り |
| ¥n | 改行 |
| ¥r | 復帰 |
| ¥s | 空白文字 |
| ¥t | タブ |
| ¥¥ | バックスラッシュ文字 |

| | |
|---|---|
| ¥*ddd* | 1 - 3 桁の 8 進数 (*d*) |
| ¥x*dd* | 1 - 2 桁の16 進数 (*d*) |
| ¥x^*c* | 文字 (*c*) で指定された制御文字 |

## B.4 ファイル名パターン

検索や置換のダイアログでは、ファイル名パターンに完全な正規表現ではなく、構文と一致する標準 UNIX 形式のワイルドカードを使用します。認識できるメタ文字は、次のとおりです。

| メタ文字 | 説明 |
|---|---|
| * | 0 文字以上の文字列を検索します。 |
| ? | 1 文字を検索します。 |
| [] | 1 文字の文字クラスを定義します。 |
| ¥ | 直前にある文字の特殊な意味を無効化します。バックスラッシュを検索するには、¥¥ と指定します。 |

## B.5 検索例

次の例では、さまざまなメタ文字を紹介します。

| 検索パターン | 説明 |
|---|---|
| ^Start | テキスト行の先頭にある "Start" という単語を検索します。 |
| End$ | テキスト行の末尾にある "End" という単語を検索します。 |
| | 注記: UNIX オプションでは、CRLF や LF などの行の終了文字は検索対象になりません。 |
| file¥.dat | テキスト行で file.dat と完全に一致する文字列を検索します。 |
| | 注記: ピリオドはメタ文字なので、. の前にエスケープ文字を使用します。 |
| file.¥.dat | メタ文字の例です。. は、有効な 1 文字を示します。このパターンを指定すると、filea.dat、fileX.dat、file9.dat のような文字列を検索できます。 |
| file..¥.dat | メタ文字は、複数回使用することができます。この例では、file、任意の 2 文字、.dat の順に並ぶ文字を含む文字列を検索することができます。 |
| file..?¥.dat | 反復用メタ文字の例です。文字 ? を指定すると、直前にある正規表現 (この場合、2 番 |

目のメタ文字 .) を含まない文字列、または、1 つ含む文字列を検索できます。この例では、file、任意の 1 文字または 2 文字、.dat の順に並ぶ文字を含む文字列を検索することができます。

`file.*¥.dat` この例では、別の反復用メタ文字を使用しています。* を指定すると、前にある正規表現 (この場合、メタ文字 .)を 0 個以上含む文字列を検索できます。この例では、file、有効な文字 (文字数の制限はありません)、.dat の順に続く文字列を検索できます。

`file[ABC]¥.dat` 文字クラスの例です。文字クラスには、有効な文字のリストが含まれます。この場合、文字 A、B、C が有効な文字として定義されています。このパターンでは、fileA.dat、fileB.dat、fileC.dat などが検索されます。

`file[0-9]¥.dat` 文字クラスでは、ハイフン (-)を使用して文字範囲を指定することができます。この例の文字クラスは、0 から 9 までの数字です。このパターンでは、file、数字、.dat の順に続く文字列を検索することができます。

`file[0-9A-F]+¥.dat` ここまでで最も複雑な例です。文字クラスには、0 から 9 までと A から F までの 2 つの範囲が指定されています。これは、16 進数を示します。メタ文字 + を使用すると、直前にある正規表現 (この場合、文字クラス) を 1 つ以上含む文字列を検索できます。このパターンでは、file、1 つ以上の 16 進数、.dat の順に続く文字列を検索できます。

`file(¥.dat)?` サブ文字列を使用すると、複数の文字を 1 つの論理正規表現にまとめることができます。この例では、パターン ¥.dat は、サブ文字列に含まれており、後にメタ文字 ? が続いています。? を使用すると、直前にある正規表現 (この場合、サブ文字列全体) を含まない文字列、または、1 つ含む文字列を検索できます。この例では、file または file.dat を検索することができます。

---

注記: サブ文字列がない場合、パターン file¥.dat? では、file.da または file.dat を検索することになります。

---

`file¥.(dat)|(idx)` この例では、サブ文字列とオプションのメタ文字 | が使用されています。オプションのメタ文字を使用すると、左側または右側の正規表現を検索できます。このパターンでは、file. の次に dat または idx が続く文字列を検索することができます。

`"file.dat"` 検索文字列を二重引用符で囲むと、二重引用符内の他のメタ文字をすべて無効にすることができます (エスケープ文字は除きます)。この例では、file.dat を検索することができます。

## B.6 置換例

正規表現の本当の威力は、検索操作で検出された文字列を置換するときに顕著に発揮されます。サブ文字列の演算子は、置換対象にフォーカスを設定する場合に不可欠です。

| 検索パターン | 置換パターン | コメント |
| --- | --- | --- |
| `"file.dat"` | `newfile` | リテラル文字列を検索し、別のリテラル文字列で置換します。 |
| `(.*)¥.htm` | `&1.html` | まず、末尾が `.htm` である文字列を検索し、次に、検索パターンと一致し、かつ、後に `.html` が続く文字列で置換します。 |
| `¥"file([0-9A-F]+)¥.dat¥"` | `"newname&1.data"` | この検索文は、上記の例を応用したものです。この文では、二重引用符内にある 16 進ベースのファイル名を検索します。二重引用符はエスケープ文字によりエスケープされています。16 進数を囲むサブ文字列の区切り文字によりフォーカスが設定されます。置換文字列は、`newname`、検索文字列の 16 進数、新しい拡張子の順に並ぶ文字列です。`"file9F.dat"` は、`"newname9F.data"` に置換されることになります。 |

# 付録C: 旧バージョンの UNIX オプションとの互換性

この章では、NetExpress Version 3 と NetExpress Version 2 で提供される UNIX オプションの互換性について説明します。次の各項では、「旧 UNIX オプション」は、NetExpress V2.0 で提供される UNIX オプションを指します。また、「新 UNIX オプション」とは、NetExpress V3.0 で提供される UNIX オプションを指します。

## C.1 設定の保存

UNIX オプションの設定は、NetExpress のプロジェクト ファイルに保存されます。NetExpress V3.0 では、これらの設定の保存形式が更新されています。UNIX オプションの設定が変更されるたびに、ファイル マッピング以外の旧 UNIX オプションの設定は、自動的に新 UNIX オプションの形式に変換され、デフォルト値が割り当てられます。

旧 UNIX オプションには、ファイルの種類という概念がなく、ファイルはすべてバイナリとして転送されます。そのため、古いファイル マップが読み込まれると、すべてのファイルに、新 UNIX オプションのデフォルトで通常割り当てられるテキストではなく、バイナリが割り当てられます。

## C.2 制約

旧 UNIX オプションでは、保存データをサーバー固有の部分とプロジェクト固有の部分に分けることはありません。そのため、次のような制約があります。

- 最終パブリッシュ日時

  旧 UNIX オプションで定義された最終パブリッシュ日時は、新 UNIX オプションでは無効になります。

- CGI サポート

  旧 UNIX オプションでは、CGI サポートはサーバーごとに定義されます。一方、新 UNIX オプションでは、プロジェクトごとに定義されます。旧 UNIX オプション サーバーから最初に読み込まれたサーバーにより、CGI サポート フラグが有効化または無効化されます。

- ファイル マッピングの有効化

  旧 UNIX オプションでは、次の操作が可能です。

  - ファイル名マッピングを有効化するか、または無効化するかを選択できます。ファイル名マッピングを無効化すると、ネイティブの PC ファイルシステム名が使用されます。

  - ファイル名マッピング情報は、プロジェクトごとに適用されますが、ファイル名マッピングの有

効化は、サーバーごとに決定します。

新 UNIX オプションでは、ファイル名マッピングを無効化することはできません。すべてのファイル名マッピング情報は、プロジェクトごとに適用されます。プロジェクト ファイルを最初に読み込んだときに、旧 UNIX オプションで使用されていたデフォルトのサーバーを基に、ファイル名マッピングが有効化されているか無効化されているかを判断します。ファイル名マッピングが有効化されている場合、既存のファイル名マッピング情報が使用されます。無効化されている場合は、ファイルシステム情報を基にファイル名マッピングが行われます。

## C.3 クライアント サーバーの互換性

旧 UNIX オプションで使用されるクライアント サーバー プロトコルは、新 UNIX オプションでは更新されています。新 UNIX オプションでは、機能が拡張され、性能が大幅に向上しています。プロトコル ストリームには、バージョンを確認して、あるプラットフォームの新しいコンポーネントと他のプラットフォームの古いコンポーネントとが相互に操作できるようにする情報が含まれています。

NetExpress V2.0 クライアントから新しい NetExpress V3.0 SCP プログラムをインストールしたサーバーへパブリッシュする場合、互換性の問題はまったくありません。

NetExpress V3.0 クライアントから以前の NetExpress V2.0 SCP プログラムをインストールしたサーバーへパブリッシュする場合、次のような制約があることにご注意ください。

- プロジェクトの論理ディレクトリを定義することができますが、論理ディレクトリに関連付けた実際のディレクトリは、[BuildDir] または [CopyDir] の実際のディレクトリと同一である必要があります。つまり、実際のターゲット ディレクトリは 2 つまで定義できることになります。
- ソース確認操作は、サポートされません。
- サーバーのディレクトリ自動作成機能は、サポートされません。
- NetExpress IDE の出力ペインには、サーバーでのビルド過程は表示されません。出力ペインは、サーバーでのビルドが完了したときだけに更新されます。
- サーバーでビルドが開始されると、パブリッシュを中止することはできません。
- パブリッシャの性能は向上しません。

# 索引