

FUJITSU Software

NetCOBOL V12.0

入門ガイド

Linux(64)

J2UL-2304-01Z0(00)
2017年12月

まえがき

本書の目的

本書は、製品の特長や主な機能について説明しています。また、COBOLアプリケーションを開発・運用するための流れを理解できるよう、簡単なサンプルアプリケーションを用いて開発から運用までの操作を説明しています。

製品で提供している開発環境・実行環境の使い方に基づいて説明していますので、本製品を初めて使用される方は本書をご一読ください。

本書の読者

本書は、本製品を初めて使用される方を対象としています。なお、本書を読むためには、以下の知識が必要です。

- ・ COBOLの文法に関する基本的な知識
- ・ 使用するOSに関する基本的な知識

本書の構成

本書は以下の構成になっています。

第1章 NetCOBOLとは

本製品の特長と主な機能、製品体系について説明しています。

第2章 NetCOBOLアプリケーションのリモート開発

小入出力機能を使うCOBOLアプリケーションを例に、NetCOBOL Studioによるリモート開発方法について説明しています。

第3章 画面帳票アプリケーションの開発

NetCOBOLシリーズのMeFt、FORMによる基本的な画面帳票アプリケーションの作成方法について説明しています。

第4章 MeFt/Web環境の構築

画面帳票アプリケーションのMeFt/Web化の方法、環境設定などについて説明しています。

第5章 効率のよいプログラムのテクニック

効率のよいCOBOLプログラムの作成テクニックについて説明しています。

第6章 サンプルプログラム

NetCOBOLが提供するサンプルプログラムについて説明しています。

登録商標について

- ・ Linuxは、Linus Torvalds氏の米国およびその他の国における登録商標あるいは商標です。
- ・ UNIXは、米国およびその他の国におけるオープン・グループの登録商標です。
- ・ Red Hat、RPM、およびRed Hatをベースとしたすべての商標とロゴは、Red Hat, Inc.の米国およびその他の国における登録商標あるいは商標です。
- ・ Microsoft、Windows、Visual Studio、およびWindows Serverは、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。
- ・ OracleとJavaは、Oracle Corporationおよびその子会社、関連会社の米国およびその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。
- ・ Adobe、Adobeロゴ、Acrobatは、Adobe Systems Incorporated (アドビシステムズ社)の商標です。
- ・ そのほか、本書に記載されている会社名および製品名は、それぞれ各社の商標または登録商標です。

製品の呼び名について

本書では、各製品を次のように略記しています。あらかじめご了承ください。

正式名称	略称
Red Hat(R) Enterprise Linux(R) 7 (for Intel64)	RHEL7
Red Hat(R) Enterprise Linux(R) 6 (for Intel64)	RHEL6
Microsoft(R) Windows Server(R) 2016 Datacenter Microsoft(R) Windows Server(R) 2016 Standard Microsoft(R) Windows Server(R) 2016 Essentials	Windows Server 2016
Microsoft(R) Windows Server(R) 2012 R2 Datacenter Microsoft(R) Windows Server(R) 2012 R2 Standard Microsoft(R) Windows Server(R) 2012 R2 Essentials Microsoft(R) Windows Server(R) 2012 R2 Foundation	Windows Server 2012 R2
Microsoft(R) Windows Server(R) 2012 Datacenter Microsoft(R) Windows Server(R) 2012 Standard Microsoft(R) Windows Server(R) 2012 Essentials Microsoft(R) Windows Server(R) 2012 Foundation	Windows Server 2012
Microsoft(R) Windows Server(R) 2008 R2 Datacenter Microsoft(R) Windows Server(R) 2008 R2 Enterprise Microsoft(R) Windows Server(R) 2008 R2 Standard Microsoft(R) Windows Server(R) 2008 R2 Foundation	Windows Server 2008 R2
Windows(R) 10 Home Windows(R) 10 Pro Windows(R) 10 Enterprise Windows(R) 10 Education	Windows 10
Windows(R) 8.1 Windows(R) 8.1 Pro Windows(R) 8.1 Enterprise	Windows 8.1
Windows(R) 7 Home Premium Windows(R) 7 Professional Windows(R) 7 Enterprise Windows(R) 7 Ultimate	Windows 7
Microsoft(R) Visual C++(R) development system	Visual C++
Microsoft(R) Visual Basic(R) programming system	Visual Basic

- ・「RHEL6」および「RHEL7」を指す場合は「Linux」または「Linux(64)」と表記します。
- ・以下をすべて指す場合は「Windows」と表記します。
 - － Windows Server 2016
 - － Windows Server 2012 R2
 - － Windows Server 2012
 - － Windows Server 2008 R2
 - － Windows 10
 - － Windows 8.1

注意事項

- ・ 本書に記載されている画面は、使用されているシステムにより異なる場合がありますので注意してください。
- ・ 本書では、“COBOL文法書”で“原始プログラム”と記述されている用語を“ソースプログラム”と記述しています。

お願い

- ・ 本書を無断で他に転載しないようお願いします。
- ・ 本書は予告なしに変更されることがあります。

輸出管理について

本ドキュメントを輸出または第三者へ提供する場合は、お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ、必要な手続きをおとりください。

2017年12月

Copyright 2014-2017 FUJITSU LIMITED

目 次

第1章 NetCOBOLとは	1
1.1 NetCOBOLの特長	1
1.2 NetCOBOLと先端技術	2
1.3 製品体系	3
第2章 NetCOBOLアプリケーションのリモート開発	6
2.1 概要	6
2.1.1 リモート開発とは	6
2.1.2 リモート開発のメリット	6
2.1.3 リモート開発の流れ	6
2.1.4 リモート開発の注意点	7
2.2 NetCOBOL Studioとは	9
2.3 作成するアプリケーションについて	10
2.3.1 アプリケーション開発の流れ	11
2.4 リモート開発によるプログラムの作成	11
2.4.1 サーバOS側の環境設定	11
2.4.2 サーバ情報の設定	12
2.4.3 プロジェクトで使用するサーバ情報の設定	16
2.4.4 メイクファイルの生成	16
2.4.5 サーバ側でビルド	17
2.4.6 サーバ側でデバッグ	18
第3章 画面帳票アプリケーションの開発	21
3.1 概要	21
3.1.1 画面帳票アプリケーションの概要	21
3.1.2 作成するアプリケーションについて	22
3.1.3 アプリケーション開発の流れ	22
3.2 表示ファイルのプログラミング	23
3.2.1 環境部(ENVIRONMENT DIVISION)	23
3.2.2 データ部(DATA DIVISION)	24
3.2.3 手続き部(PROCEDURE DIVISION)	25
3.2.3.1 画面機能	25
3.2.3.2 帳票機能	26
3.2.4 エラー処理	26
3.3 画面帳票定義体の作成	27
3.3.1 画面定義体の作成	27
3.3.1.1 FORMの起動	28
3.3.1.2 画面定義体の属性設定	28
3.3.1.3 項目の配置と属性、レコード情報、罫線情報の設定	29
3.3.1.4 アテンション情報の設定	35
3.3.1.5 定義の正当性の確認	37
3.3.1.6 画面定義体の保存	37
3.3.2 帳票定義体の作成	37
3.3.2.1 PowerFORMの起動	38
3.3.2.2 項目の配置と属性設定	39
3.3.2.3 繰返しの設定	40
3.3.2.4 罫線の定義	42
3.3.2.5 帳票定義体の保存	44
3.4 プログラムの翻訳とリンク	44
3.5 MeFt/Web環境の構築	45
3.6 実行	45
第4章 MeFt/Web環境の構築	46
4.1 概要	46
4.1.1 MeFt/Webの概要	46
4.1.2 構築作業の流れ	47

4.2 MeFt/Webサーバのセットアップ	47
4.2.1 Webサーバの環境設定	48
4.2.1.1 仮想ディレクトリ(エイリアス)の設定	48
4.2.1.2 実行時ユーザの設定	49
4.2.1.3 Webサーバの再起動時	49
4.2.2 MeFt/Web動作環境の設定	49
4.2.3 利用者プログラムの指定	50
4.2.4 MeFt/Web サーバの起動	51
4.3 MeFt/Web クライアントのセットアップ	51
4.3.1 MeFt/Webコントロールのダウンロード	51
4.4 ウィンドウ情報ファイル、プリンタ情報ファイルの準備	52
4.5 HTMLの作成	52
4.6 リモート実行	55
4.6.1 HTMLの表示	55
4.6.2 プログラムの実行	57
4.7 通信が切断されるパターンについて	61
第5章 効率のよいプログラムのテクニック	63
5.1 一般的なテクニック	63
5.1.1 作業場所節の項目	63
5.1.2 ループの最適化	63
5.1.3 複合条件の判定順序	63
5.2 データ項目の属性を理解して使う	63
5.2.1 英数字項目と数字項目	63
5.2.2 USAGE DISPLAYの数字項目(外部10進項目)	63
5.2.3 USAGE PACKED-DECIMALの数字項目(内部10進項目)	64
5.2.4 USAGE BINARY/COMP/COMP-5の数字項目(2進項目)	64
5.2.5 数字項目の符号	64
5.3 数字転記・数字比較・算術演算の処理時間を短くする	64
5.3.1 属性	64
5.3.2 桁数	64
5.3.3 べき乗の指数	64
5.3.4 ROUNDED指定	64
5.3.5 ON SIZE ERROR指定	65
5.3.6 TRUNCオプション	65
5.4 英数字転記・英数字比較を効率よく行う	65
5.4.1 境界合せ	65
5.4.2 項目長	65
5.4.3 転記の統合	65
5.5 入出力におけるテクニック	65
5.5.1 SAME RECORD AREA句	65
5.5.2 ACCEPT文、DISPLAY文	65
5.5.3 OPEN文、CLOSE文	66
5.6 プログラム間連絡におけるテクニック	66
5.6.1 副プログラムの分割の基準	66
5.6.2 動的プログラム構造と動的リンク構造	66
5.6.3 CANCEL文	66
5.6.4 パラメタの個数	66
5.7 デバッグ機能を使用する	66
5.8 数字項目の標準規則	67
5.8.1 10進項目	67
5.8.2 2進項目	67
5.8.3 浮動小数点項目	68
5.8.4 乗除算の混合時の小数部桁数	68
5.8.5 絶対値がとられる転記	68
5.9 注意事項	68
第6章 サンプルプログラム	69

6.1 標準入出力を使ったデータ処理(sample01).....	69
6.2 行順ファイルと索引ファイルの操作(sample02).....	70
6.3 表示ファイル機能を使ったプログラム(sample03).....	71
6.4 COBOLプログラム間の呼出し(sample04).....	73
6.5 コマンド行引数の受取り方(sample05).....	75
6.6 環境変数の操作(sample06).....	76
6.7 印刷ファイルを使ったプログラム(sample07).....	77
6.8 印刷ファイルを使ったプログラム(応用編)(sample08).....	79
6.9 FORMAT句付き印刷ファイルを使ったプログラム(sample09).....	81
6.10 オブジェクト指向プログラム(初級編)(sample10).....	83
6.11 データベース機能を使ったプログラム(sample16).....	84
6.12 データベース機能を使ったプログラム(応用編)(sample17).....	87
6.13 COBOLファイルアクセスルーチンのサンプル.....	90
6.13.1 行順ファイルの読み込み.....	90
6.13.2 行順ファイルの読み込みと索引ファイルの書出し.....	90
6.13.3 索引ファイルの情報の取得.....	91
索引.....	93

第1章 NetCOBOLとは

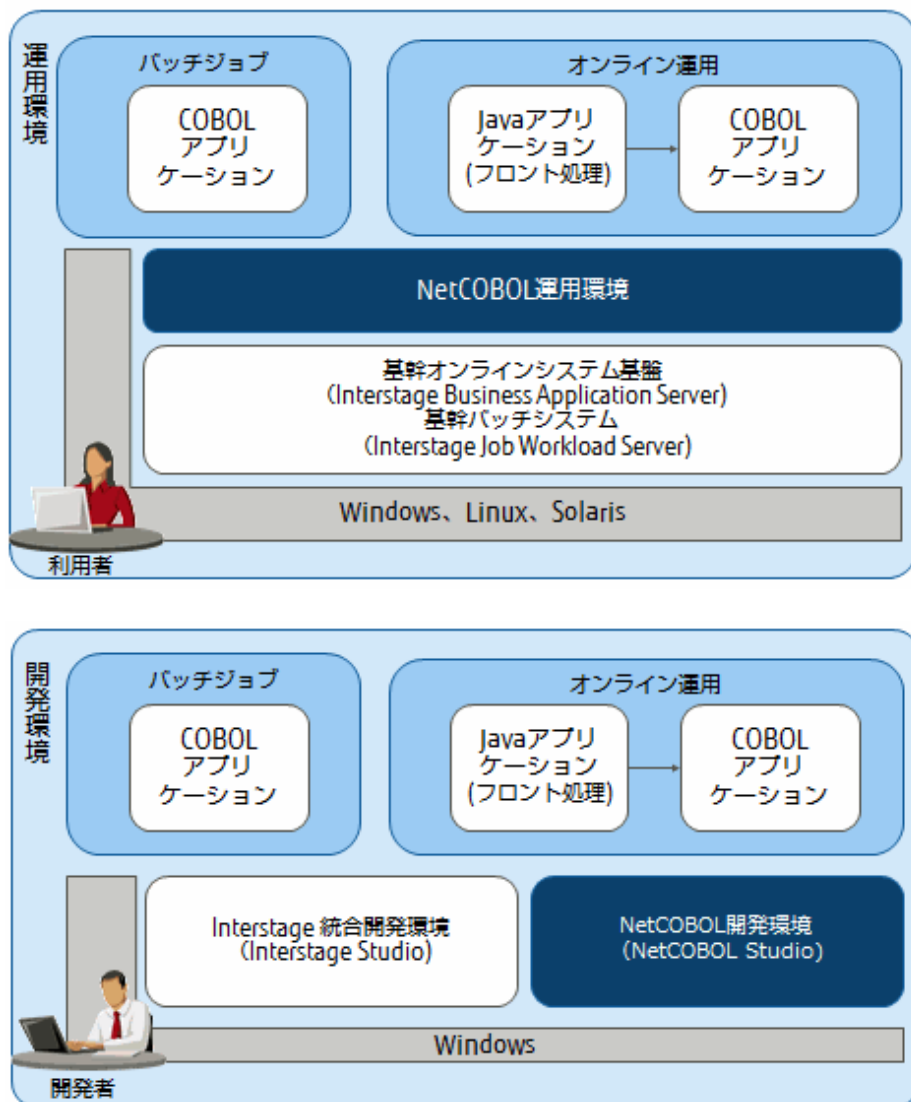
COBOL言語は、1960年に誕生して以来、ビジネスロジックの記述性や互換性などに優れている点が評価され、多くのビジネスシステムに使用され続けています。一方、ITの進展に伴い、ビジネスシステムの要件が急速に高度化・多様化しています。

COBOLは、その長い歴史において、常に最新テクノロジーや時代の要求に応じて進化してきました。

お客様の既存COBOL資産を活かし、長期に渡り安定してビジネスの成長を支援するのが「NetCOBOL」です。

クラウドやビッグデータ活用を支える富士通のソフトウェア製品と組み合わせることで、お客様のCOBOL資産の価値をさらに高めます。

Windows、Linux、Solaris 開発・運用環境



ここでは、NetCOBOLシリーズの概要について述べています。ご利用のシステムによってはサポートされない機能や連携製品が含まれます。ソフトウェア説明書をご確認ください。

1.1 NetCOBOLの特長

COBOL (COmmon Business Oriented Language) は、事務処理向けに開発されたプログラム言語です。10進演算、ファイル処理、帳票作成などの事務処理に特化した仕様と英語表現に似た文法で、読み易く分かり易いプログラム記述が可能です。

NetCOBOLは、COBOLの特長を活かし、最新テクノロジー・最新環境に対応したオープンプラットフォームのCOBOL開発環境です。

NetCOBOLには、以下の特長があります。

COBOL資産を長期間、安心して利用

国際規格、業界標準仕様に対応し、上位との互換性も保証しています。将来も安心できる基幹システム運用と拡張が可能です。メインフレームやオフコンの既存COBOL資産と開発者のスキル、ノウハウも活用できます。

効率的で高生産なプログラム開発

COBOL統合開発環境により、設計・プログラム・テスト・保守まで、開発プロセス全体を効率化できます。バッチ、Webアプリケーションからクラウドアプリケーションまで、最新技術と連携したプログラム開発が可能です。

基幹システムの適用範囲拡大

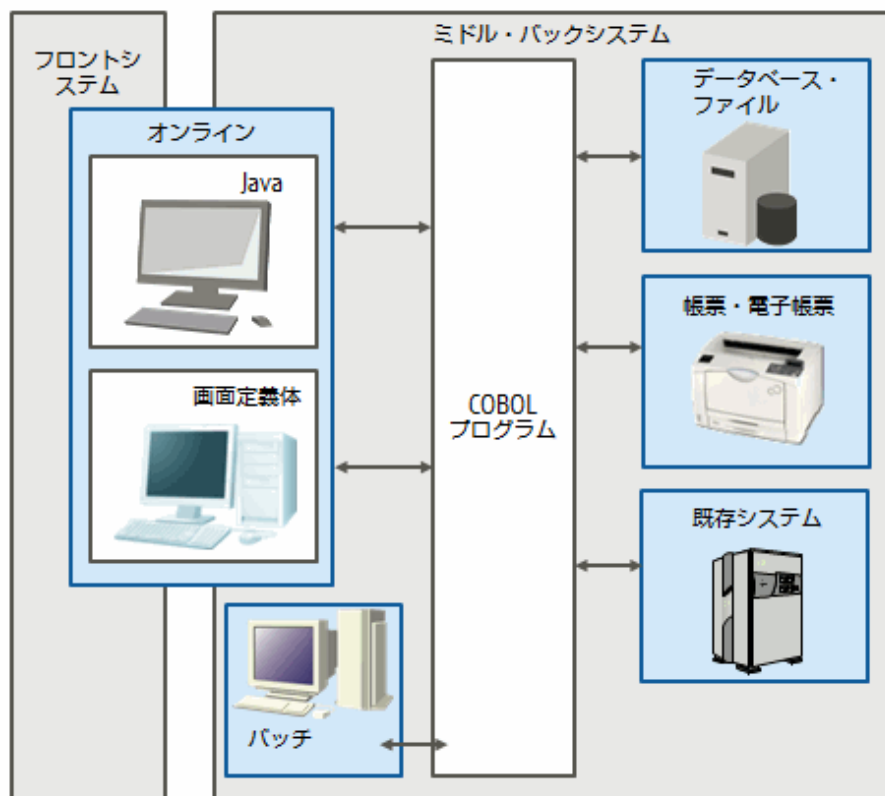
.NET、Java、クラウド連携で、基幹システムの適用範囲を拡大できます。特に基幹システムを支える富士通のソフトウェア「Interstage」との連携により、堅牢で柔軟性の高い基幹システムを構築可能です。

高い実績と安心サポート

メインフレーム、オフコンのCOBOLは50年、オープンプラットフォームのCOBOLは20年の実績を持ち、富士通の支援サービス「SupportDesk」と全国各地の営業拠点のサポート体制により、安心して利用できます。

1.2 NetCOBOLと先端技術

NetCOBOLは、COBOLの国際規格をベースに、各種RDB、画面・帳票、最新インターネット技術を組み合わせたCOBOLアプリケーションを作成することができます。NetCOBOLなら、変化の激しい今日のビジネス環境で、お客様の経営を支える基幹システムの信頼性、安定性、効率化を迫及したシステム構築が可能です。



オンライン

Web技術、柔軟性の高いJavaでフロントシステムを利用し、ビジネスロジックの生産性、実行性能の高いCOBOLを、ミドル・バックシステムに利用することで、言語特性を利用し、拡張性の高い基幹システム構築が可能です。フロント、ミドル・バックシステムの間、富士通のアプリケーションサーバ「Interstage Application Server」を配置することにより、堅牢なトランザクションシステムを構築できます。更に、「Interstage Business Application Server」を利用することで、高度な制御ロジックを実現できます。

バッチ

膨大なビジネスデータを扱うバッチ業務は、基幹システムを支える基盤です。メインフレーム・オフコンの富士通COBOLから継承された、NetCOBOLの高い実行性能、信頼性は、オープンシステムのバッチ業務に最適です。さらにバッチ処理基盤「Interstage Job Workload Server」を組み合わせることで、バッチ処理の安定稼動と運用性向上を実現することができます。また、総合運用管理「Systemwalker」との連携で、サーバの起動、終了からバッチ処理の起動、エラーリカバリーまで、24時間365日、トータルなバッチ運用を実現できます。

また、Linux(64)版 NetCOBOL Enterprise Editionでは、Apache Hadoopおよび当社の並列分散処理ソフトウェア「Interstage Big Data Parallel Processing Server」と連携し、並列分散処理によるCOBOLバッチ処理の高速化を実現できます。

データベース・ファイル

基幹システムの中核となるのは、データ集計・分析です。ビジネスの拡大に伴い、取り扱うデータも飛躍的に増加しています。NetCOBOLと各種RDBを組み合わせることにより、データの集計・分析をより早く実現できます。さらに「PowerSORT」を導入することにより、データのソート・マージが高速化され、基幹システムのパフォーマンスを向上できます。

画面定義体

入力データチェック、ファンクションキー、カーソル移動など、基幹システムの画面に求められる操作が可能です。

Linux(64)版 NetCOBOLでは、この画面定義体をMeFt/Webを使用して利用することが可能です。

メインフレーム・オフコンなど従来システムと同様の画面操作を、オープンシステムで実現できるため、システムをご利用になるお客様の教育も必要ありません。画面は、NetCOBOLの専用ツール「FORM」で作成、COBOLプログラムからは、READ文/WRITE文を利用することで、COBOLのノウハウで画面操作性の優れた基幹システムを構築できます。

帳票・電子帳票

基幹システムに求められる、きめ細かな帳票出力を実現できます。豊富な文字、罫線、図形により、きれいで見やすい帳票が作成できます。帳票製品「Interstage List Works」と連携することで、COBOLプログラムからの帳票出力をそのまま電子化できます。これにより、業務システムのコスト削減、帳票情報の共有が図れます。帳票はNetCOBOLの専用ツール「FORM」または「PowerFORM」で作成、COBOLプログラムからの帳票出力は、WRITE文を利用し、COBOLのレコード出力イメージで帳票出力が可能です。

既存システム

メインフレーム・オフコンの基幹システムをご利用ならば、オープンシステムをアドオンし、トータルなシステム構築も可能です。メインフレーム・オフコンは、長期間、高可用性が求められる業務に最適であり、オープンシステムは、即時性、定期的に見直しが可能な業務に最適です。メインフレームとは、富士通のアプリケーションサーバ「Interstage」によるアプリケーション間の連携が可能です。オフコンとは、データベースの複製・共用管理「PowerReplication」を利用したデータ連携が可能です。

1.3 製品体系

開発・運用環境製品は、複数のコンポーネントで構成されています。インストール時にカスタムインストールを選択して、必要なコンポーネントだけをインストールすることもできます。

インストールの詳細は、“ソフトウェア説明書”を参照してください。

以下の表の記号の意味は以下のとおりです。

EE : Enterprise Edition

SE : Standard Edition

BE : Base Edition

○ : 製品に同梱されるコンポーネント

× : 製品に同梱されないコンポーネント

表1.1 開発・運用パッケージ

コンポーネント名	機能名	EE	SE	BE
NetCOBOL	COBOLコンパイラ	○	○	○
	COBOLランタイム	○	○	○
	Hadoop連携機能	○	×	×
Jアダプタクラスジェネレータ	Java連携(注3)	○	○	×

コンポーネント名	機能名	EE	SE	BE
MeFt	帳票の運用	○	○	×
MeFt/Web	Webアプリケーションの構築支援	○(注2)	○(注2)	×
MeFt/Web HTML変換方式	画面定義体のWebコンテンツ(HTML)変換	○	○	×
Migration CJC for INTARFRM 連携機能 (注1)	COBOL-Java(Servlet/JSP)移行支援	○	○	×
SIMPLIA/TF-MDPORT	開発資産流用支援	○	×	×
SIMPLIA/TF-LINDA	テストデータ作成・更新・検証支援	○	×	×
富士通メインフレーム浮動小数点演算エミュレータ	富士通メインフレーム形式の浮動小数点データ	○	×	×
PowerSORT	高性能データ・ソートマージ	○	×	×

表1.2 運用パッケージ

コンポーネント名	機能名	EE	SE	BE
NetCOBOL	COBOLランタイム	○	○	○
	Hadoop連携機能	○	×	×
Jアダプタクラスジェネレータ	Java連携(注3)	○	○	×
MeFt	帳票の運用	○	○	×
MeFt/Web	Webアプリケーションの構築支援	○(注2)	○(注2)	×
MeFt/Web HTML変換方式	画面定義体をWebコンテンツ(HTML)変換	○	○	×
Migration CJC for INTARFRM 連携機能 (注1)	COBOL-Java(Servlet/JSP)移行支援	○	○	×
富士通メインフレーム浮動小数点演算エミュレータ	富士通メインフレーム形式の浮動小数点データ	○	×	×
PowerSORT	高性能データ・ソートマージ	○	×	×

注1 : Migration CJC for INTARFRMサービスの契約が必要です。

注2 : MeFt/Webアプリケーションの利用時には、以下のソフトウェアが必須です。

- ・ サーバ側
Interstage Application Server
- ・ クライアント側
Microsoft Internet Explorer (32bit版)

注3 : Linux(64)版の富士通製JDK/JDKまたはOracle製JDK/JREが別途必要です。

必須ソフトウェアの詳細は、“NetCOBOL ソフトウェア説明書”を参照してください。



注意

Linux(64)版 NetCOBOL シリーズでは、画面帳票設計ツール(FORM、FORMオーバーレイオプションおよびPowerFORM)を提供していません。帳票を作成する場合は、Windows版の画面帳票設計ツール(FORM、FORMオーバーレイオプションおよびPowerFORM)をお使いください。

第2章 NetCOBOLアプリケーションのリモート開発

本章では、NetCOBOLが提供するリモート開発の機能を説明するとともに、簡単なアプリケーションの作成を通じて開発環境の操作を説明します。

なお、ここで説明するNetCOBOL Studioを使用したリモート開発を行うためには、Windows版 NetCOBOL開発パッケージが別途必要です。

2.1 概要

2.1.1 リモート開発とは

リモート開発を行うことで、広く普及しているWindowsシステムを活用して、COBOLアプリケーションを効率よく開発することができます。

リモート開発を行うには、NetCOBOL Studioを含むNetCOBOL開発系製品をインストールしたシステムが別途必要になります。ここでは、Linux(64)版のNetCOBOLがインストールされたシステムをリモート開発の「サーバ側」と呼び、NetCOBOL Studioがインストールされたシステムをリモート開発の「クライアント側」と呼びます。

リモート開発では、開発者はクライアント側のNetCOBOL Studioを用いて作業を行います。NetCOBOL Studioは、必要があればサーバ側に接続し、サーバ側で翻訳などの開発作業を行い、その結果をNetCOBOL Studioに表示します。

2.1.2 リモート開発のメリット

COBOLアプリケーションの多くは、高価なサーバマシンで運用されます。これらのCOBOLアプリケーションを同様のシステム上で開発する場合、以下の問題があります。

- ・ これらのシステムではGUIベースの環境が用意されていない場合が多く、コマンドラインを用いて開発作業を行う必要があります。
- ・ マシンが貴重であり、複数の開発者がマシンを共有する必要があります。

一方、Windowsシステムは個人用端末として広く普及しており、これを利用するとGUIベースの環境を開発者が占有することができます。

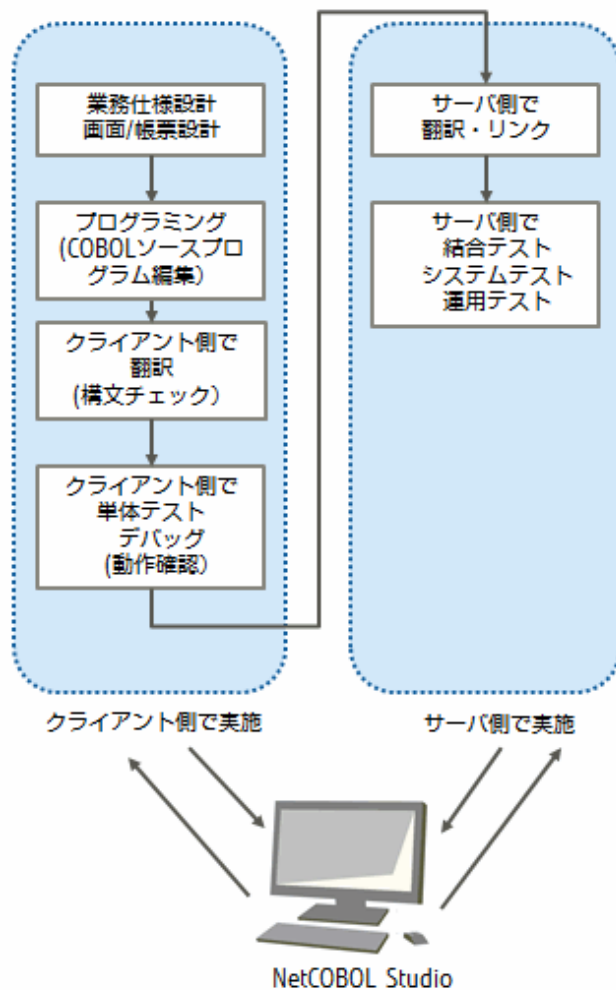
リモート開発では、開発作業をなるべくWindowsシステムで行うようにすることで、上記の問題点を解決します。

- ・ GUIベースの開発環境を用いて、開発を効率よく行うことができます。
- ・ COBOLソースプログラムの編集など可能な作業をクライアント側で行うことで、貴重なサーバ側マシンの負荷を減らします。

2.1.3 リモート開発の流れ

リモート開発の流れは以下のとおりです。

図2.1 リモート開発作業の流れ



まず、クライアント側でNetCOBOL Studioを使ってプロジェクトを作成し、COBOLソースプログラムの編集を行います。

可能であれば、翻訳、単体テスト・デバッグをクライアント側で行います。

その後、NetCOBOL Studioのリモートビルド・リモートデバッグ機能を用いてサーバ側でアプリケーションの翻訳、テスト・デバッグを行います。

2.1.4 リモート開発の注意点

一般に、COBOLプログラムは高い移植性を持っており、多くの場合、プログラムの作成から単体テストまでをクライアント側で行うことができます。

ただし、サーバ側固有の機能を利用した場合や、対象プラットフォームによって動作の異なる機能を利用した場合は、サーバ側で動作を確認する必要があります。

図2.2 UNIX系システムとWindows系システムの機能範囲

【Windows 系システムの機能範囲】		【UNIX 系システムの機能範囲】	
《国際規格 COBOL85》 - 中核 *** - 順ファイル *** - 相対ファイル *** - 索引ファイル *** - プログラム間連絡 *** - 整列併合 *** - 原始文操作 *** - 報告書作成 ** - デバッグ *** - 区分化 *** - データベース操作(SQL) *		《国際規格 COBOL2002》 - オブジェクト指向機能 *** - 利用者定義型 *** - Unicodeサポート * - 2進データ型(BINARY-SHORT 等)***	
《XPG仕様》 - 行順ファイル *** - C言語間結合 (値渡し、復帰値) *** - ファイル共用/レコード排他 *** - スクリーン操作 ** - コマンド行引数操作 *** - 環境変数操作 *** - 連結式 ***		《MicroFocus 互換仕様》 - スクリーン操作 ** - 名前付き定数(78レベル) *** - 16進数定数 ***	
《国際規格 1989 年追補》 - 組み込み関数 ***		《富士通拡張仕様》 - 表示ファイル(画面、帳票) * - 表示ファイル(APL, ACM) ** - 日本語プログラミング *** - 日本語処理 * - 拡張日本語印刷 * - ビット操作 *** - 浮動小数点数操作 *** - FORMAT 句付き印刷ファイル (順ファイル) *** - 拡張索引ファイル (多重キー、逆順検索) *** - 定数節 *** - システムプログラム記述(SD) *** - マルチスレッド対応 *** - Web連携機能 *	
		- OOM連携機能	

記号の説明
 *** : 共通機能範囲
 ** : Linux 版 Solaris(64)版では非サポート
 * : Windows 系-UNIX 系間で機能差あり

対象プラットフォームによって動作の異なる機能には以下のものが含まれます。

日本語定数/日本語16進定数

クライアント側とサーバ側で文字コードが異なる場合、片方で翻訳可能な値が他方では翻訳エラーになる場合があります。

文字比較/索引ファイルのキー順序/ソートキーの順序

クライアント側とサーバ側での実行時文字コードの違いによって、半角カナ文字、日本語文字の大小順序の結果が異なる場合があります。

日本語字類条件

クライアント側とサーバ側での実行時文字コードの違いによって、判定結果が異なる場合があります。

拡張日本語印刷

使用できる文字、書体などは対象プラットフォームによって異なります。

データベース機能

使用するデータベース製品の違いによって、実行結果に違いが出る場合があります。

Web連携機能

本システムでは、MeFi/Webを利用したアプリケーション開発は可能ですが、COBOL Webサブルーチンを利用したアプリケーションの開発はサポートしていません。

2.2 NetCOBOL Studioとは

NetCOBOL Studioは、Windows版 NetCOBOL開発パッケージに同梱されている、Eclipse(エクリプス)をベースとしたCOBOL開発環境です。

Eclipseは、いろいろなツールをプラグインで追加していくことができるオープンソースの統合開発環境(IDE)です。特にJavaの開発環境として普及しています。

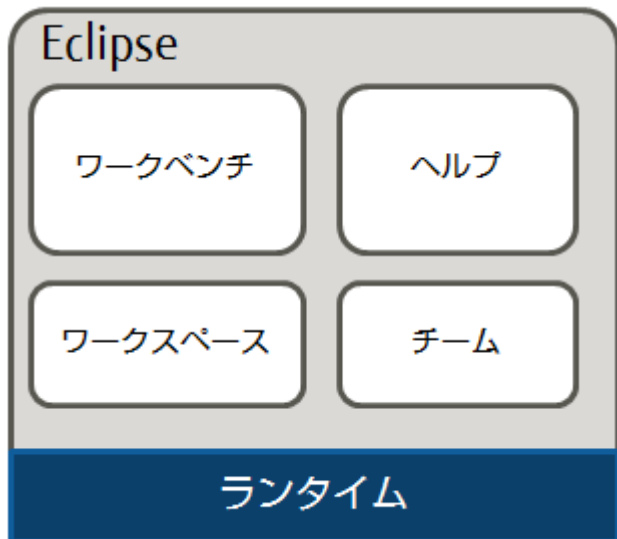
NetCOBOL Studioでは、COBOLアプリケーションの開発に必要な各種操作(プログラムの編集・翻訳・リンク・実行・デバッグ)を行うための操作ビューを持ち、プログラムの作成からデバッグまで一連の作業をサポートします。

NetCOBOL StudioはEclipseベースの開発環境です。そのため、Eclipseの解説を交えながら、NetCOBOL Studioを使用したプログラムの作成方法を説明します。

Eclipseについて

Eclipse自体は部品を入れる箱のようなもので、様々な部品を追加する(プラグインする)ことで拡張可能な機構を持つ開発環境ツールのプラットフォームです。Eclipseの基本セットは、以下のような部品から構成されています。これらの部品はEclipseのランタイムエンジンの上に乗っています。

NetCOBOLでは、EclipseにCOBOLの機能をプラグインしてCOBOL統合開発環境NetCOBOL Studioを提供しています。



ワークベンチについて

「ワークベンチ」は、開発環境のユーザインターフェースのことで、NetCOBOL Studioを起動した際に表示される画面そのものを指します。エディタやビュー、メニューなどいろいろなGUI部品を管理します。

ワークスペースについて

「ワークスペース」は後述する格納場所や依存関係などプロジェクトの情報を管理します。

プロジェクトについて

「プロジェクト」には、以下の種類があります。

- COBOLソリューションプロジェクト

複数のプロジェクト(COBOLプロジェクト、COBOLリソースプロジェクト)をまとめて管理する場合に使用します。共通オプションの設定やプロジェクトに対しての一括操作ができます。

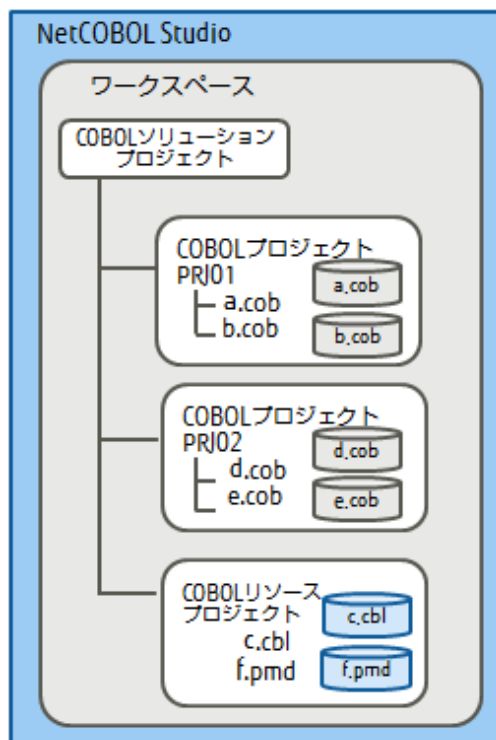
- COBOLプロジェクト

COBOLアプリケーションを作成するために使用します。プロジェクトの作成単位は、ロードモジュール単位になります。

- COBOLリソースプロジェクト

COBOL資産(登録集ファイルや定義体ファイル)の保管庫として利用します。

NetCOBOL Studio上では、プロジェクト名をトップレベルにCOBOLソースプログラムや登録集がツリー構造で表示されます。



パースペクティブについて

NetCOBOL Studioの画面は、複数の情報表示ビューから構成されます。このような情報表示ビューの組み合わせ(レイアウト)は「パースペクティブ」といいます。

COBOLプログラムの開発に最適な情報表示ビューの組み合わせを「COBOLパースペクティブ」といい、プログラムデバッグに最適な情報ビューの組み合わせを「デバッグパースペクティブ」といいます。作業内容に合わせ、それぞれのパースペクティブを利用することで、COBOLプログラムを効率的に開発・デバッグすることが可能です。

「COBOLパースペクティブ」と「デバッグパースペクティブ」の詳細は、「NetCOBOL Studio ユーザーズガイド」を参照してください。

2.3 作成するアプリケーションについて

本章では、NetCOBOLに同梱されているサンプルの中で、COBOLの小入出力機能を使って、標準入力からデータを入力したり標準出力にデータを出力したりするサンプルを使用し、アプリケーションを作成します。

アプリケーションの概要

ACCEPT文およびDISPLAY文を使用して、標準入力からアルファベット1文字(小文字)を入力し、入力したアルファベットで始まる単語を標準出力に出力します。

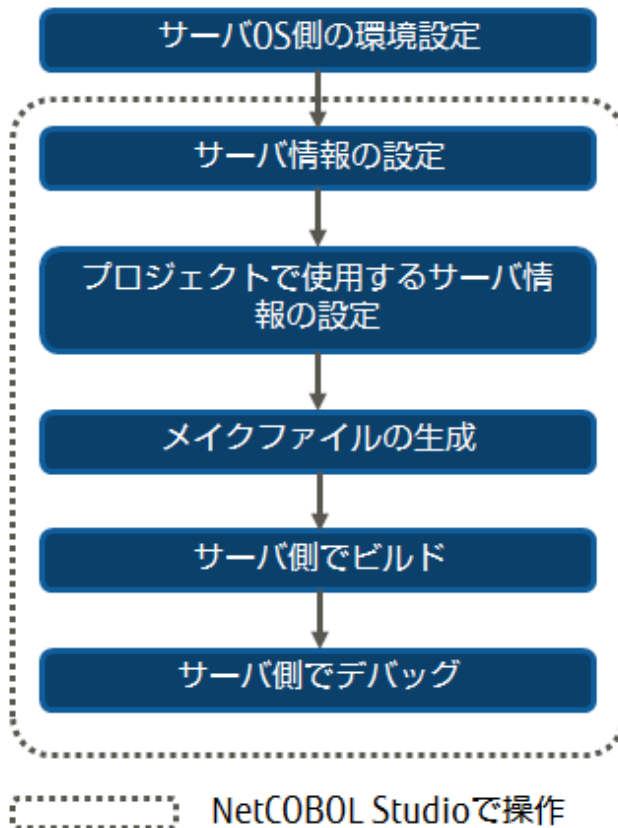
プログラムの格納場所

説明に使用するサンプルは、次の場所の、各ロケール名のディレクトリに格納されます。コピーしてご利用ください。

```
/opt/FJSCbl64/samples
```

2.3.1 アプリケーション開発の流れ

本章で説明するリモート開発の流れを次に示します。



2.4 リモート開発によるプログラムの作成

ここではローカル開発用のCOBOLプロジェクトをリモート開発用のCOBOLプロジェクトとして利用し、サーバ側のCOBOLプログラムをリモート開発する手順を説明します。

2.4.1 サーバOS側の環境設定

リモート開発を行うにはサーバ側 (Linuxシステム) の環境設定が必要です。以下を行います。

NetCOBOLリモート開発サービスの起動

リモート開発サービスを開始するには、管理者アカウントで以下のシェルスクリプトを実行します。

```
$ /opt/FJSVXrds/bin/enable-rds.sh
```

詳細は、“NetCOBOLユーザズガイド”の“リモート開発サービスの開始・停止方法”を参照してください。

環境変数の設定

ここでは、設定する環境変数が以下であると仮定して、シェルスクリプトを利用して環境変数を設定します。

- 資産の転送時に関係する環境変数
サーバ側のInterstage Charset Managerを使用してプログラム資産の送受信に必要なコード変換を行う。
- ビルドに関係する環境変数
 - COBOLプログラムの翻訳・リンクに必須の環境変数は、NetCOBOLから提供されているシェルスクリプトで設定する。

- ー 開発対象のプログラムが使用する文字コードはUnicodeとする。
- ー COBOLソースの翻訳リストは、共通のディレクトリ(../list)に保存する。
- ー 開発者が共通して参照する登録集の格納ディレクトリ(../COPYLIB)を指定する。

各環境変数の詳細については、“NetCOBOL ユーザーズガイド”または“NetCOBOL Studio ユーザーズガイド”の“サーバの環境設定”を参照してください。

シェルスクリプトの例

Linuxサーバを使用してリモート開発をする場合、ログインシェルとしてcshまたはbashを使用できます。

ログインシェルとしてcshを使用する場合、各開発者の使用するホームディレクトリにある“.cshrc”に以下のテキストを追加してください。

“.cshrc”の例

```
## COBOL環境設定
source /opt/FJSVcb164/config/cobol.csh
## Interstage Charset Managerのための環境設定
if (${?LD_LIBRARY_PATH}) then
setenv LD_LIBRARY_PATH /opt/FSUNiconv/lib:${LD_LIBRARY_PATH}
else
setenv LD_LIBRARY_PATH /opt/FSUNiconv/lib
endif
## 開発者共通の翻訳・リンク時設定
setenv COBOLOPTS "-dp ../list"
setenv COBCOPY ../COPYLIB:${COBCOPY}
## 開発対象プログラムの使用する文字コード
setenv LANG ja_JP.UTF-8
```

ログインシェルとしてbashを使用する場合、各開発者の使用するホームディレクトリにある“.bashrc”に以下のテキストを追加してください。

“.bashrc”の例

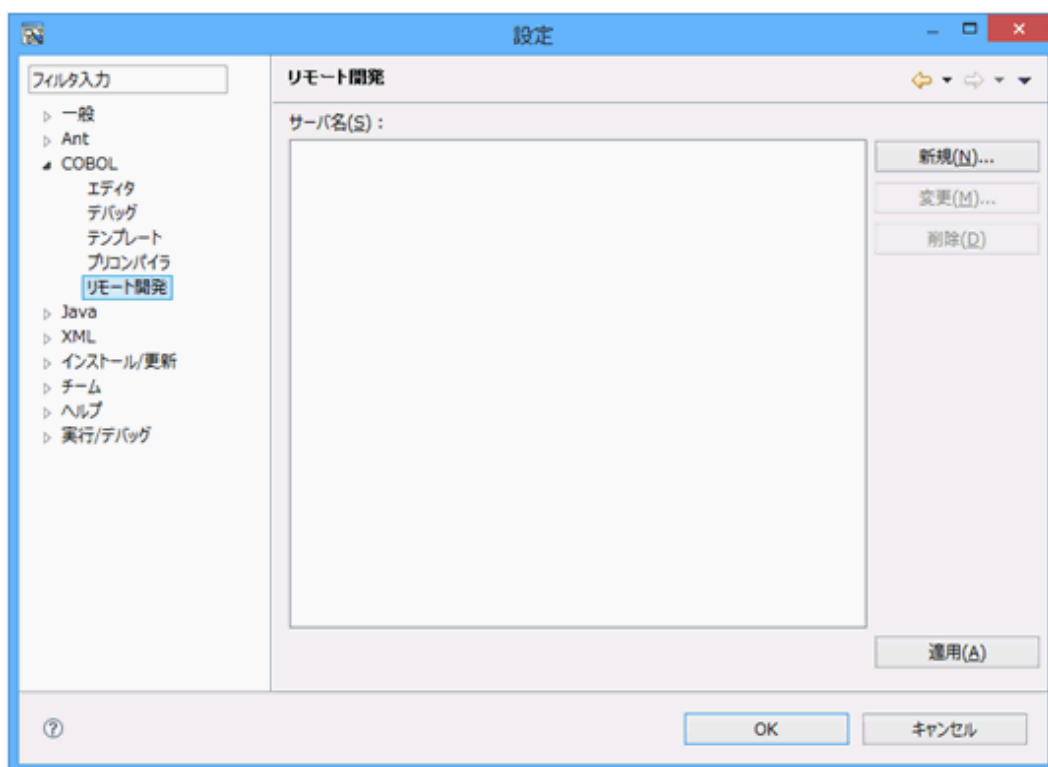
```
## COBOL環境設定
source /opt/FJSVcb164/config/cobol.sh
## Interstage Charset Managerのための環境設定
if [ ${LD_LIBRARY_PATH:-""} = "" ] ; then
LD_LIBRARY_PATH=/opt/FSUNiconv/lib; export LD_LIBRARY_PATH
else
LD_LIBRARY_PATH=/opt/FSUNiconv/lib:${LD_LIBRARY_PATH};export LD_LIBRARY_PATH
fi
## 開発者共通の翻訳・リンク時設定
COBOLOPTS="-dp ../list";export COBOLOPTS
COBCOPY=../COPYLIB:${COBCOPY}; export COBCOPY
## 開発対象プログラムの使用する文字コード
LANG= ja_JP.UTF-8; export LANG
```

2.4.2 サーバ情報の設定

サーバと連携するための情報を、以下の手順で設定します。

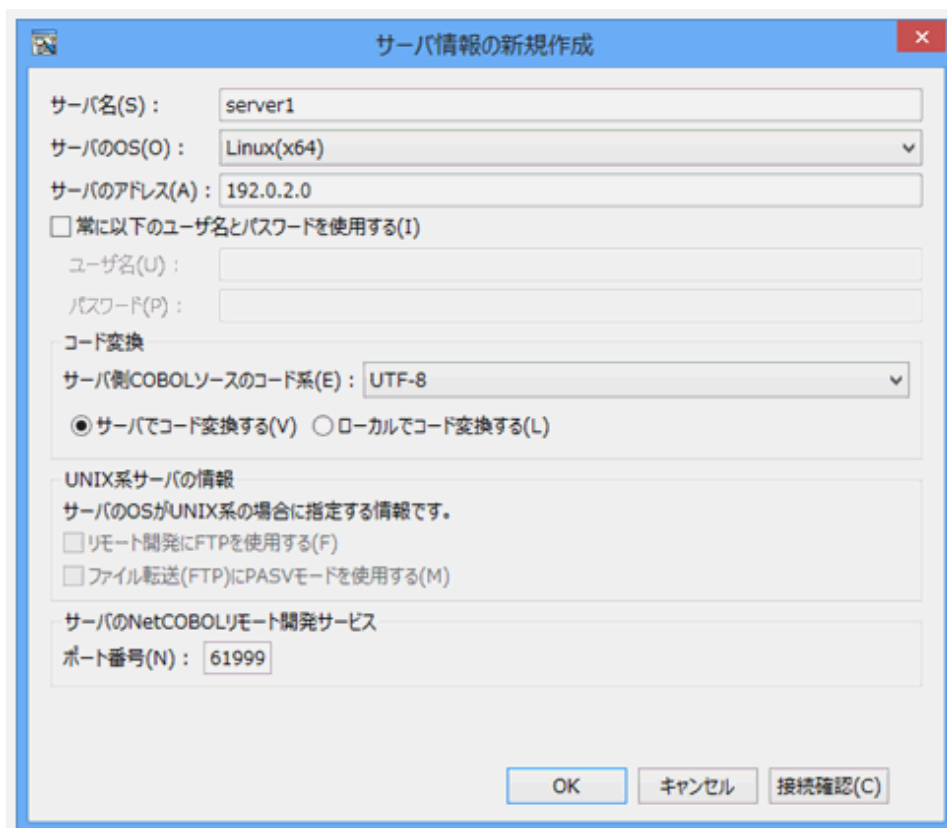
1. メニューバーから[ウィンドウ] > [設定]を選択します。
→ [設定]ダイアログボックスが表示されます。
2. 左のペインから[COBOL] > [リモート開発]を選択します。
→ [リモート開発]ページが表示されます。

3. [リモート開発]ページの[新規]ボタンをクリックします。



→ [サーバ情報の新規作成]ダイアログボックスが表示されます。

4. ここでは以下の情報を設定します。



設定項目		設定内容
サーバ名		サーバ情報を管理するための任意の名前を設定します。
サーバのOS		リモート開発するサーバのOSを選択します。
サーバのアドレス		ネットワーク上のサーバを識別するための名前 (FQDN: Fully Qualified Domain Name) またはIPアドレスを設定します。
常に以下のユーザ名とパスワードを使用する		チェックします。 このダイアログボックスで設定したユーザ名とパスワードを使用します。
	ユーザ名	サーバで使用するアカウントのユーザ名を設定します。
	パスワード	ユーザ名に付与されたパスワードを設定します。
コード変換		テキストファイルのコード変換の情報です。
	サーバ側COBOLソースのコード系	リモート開発のサーバ側に転送されたCOBOLソースのコード系を選択します。
	サーバでコード変換する	“サーバでコード変換する”を選択します。 サーバ側でコード変換処理を実行します。
	ローカルでコード変換する	[サーバのOS]で"Windows(Itanium)"または"Windows(x64)"を選択している場合は無効となります。
UNIX系サーバの情報		[サーバのOS]において、Solaris、Solaris(64)、Linux(x86)、Linux(Itanium)、Linux(x64)を選択したときに指定する情報です。
	リモート開発にFTPを使用する	リモート開発のサーバ側のサービスとしてftpd/rexecサービスを使用する場合に選択してください。
	ファイル転送(FTP)にPASVモードを使用する	PASVモードでファイル転送する場合に選択してください。 サーバ側のftpd/rexecサービスを使用するリモート開発の場合に有効となります。
サーバのNetCOBOLリモート開発サービス		サーバ側のNetCOBOLリモート開発サービスの情報です。
	ポート番号	NetCOBOLリモート開発サービスのTCP/IPのポート番号を指定します。 サーバ側のNetCOBOLリモート開発サービスを使用するリモート開発の場合に有効となります。

5. 必要な情報を設定したら[接続確認]ボタンをクリックします。

→ 設定した情報が正しいければ、[確認]シートが表示され、サーバの環境変数の情報が表示されます。

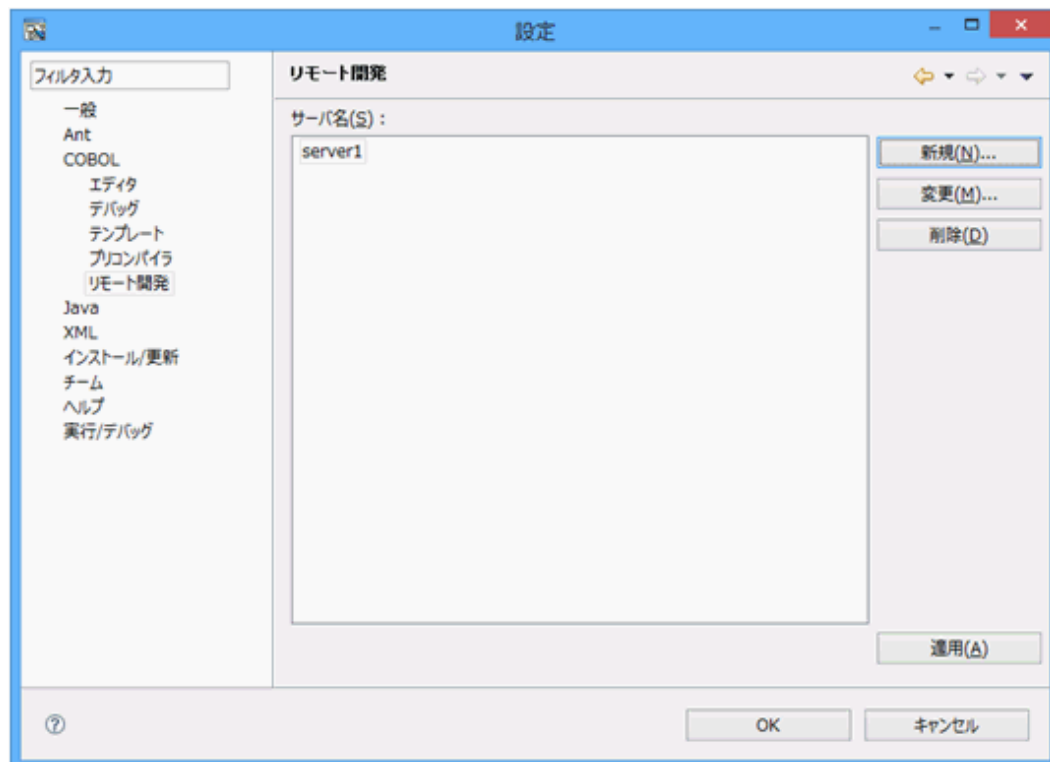


6. [OK]ボタンをクリックします。

→ [サーバ情報の新規作成]ダイアログボックスに戻ります。

7. [サーバ情報の新規作成]ダイアログボックスで[OK]ボタンをクリックします。

→ [設定]ダイアログボックスに戻り、[リモート開発]ページの[サーバ名]に[サーバ情報の新規作成]ダイアログボックスで設定したサーバ名が表示されます。



8. [OK]ボタンをクリックします。

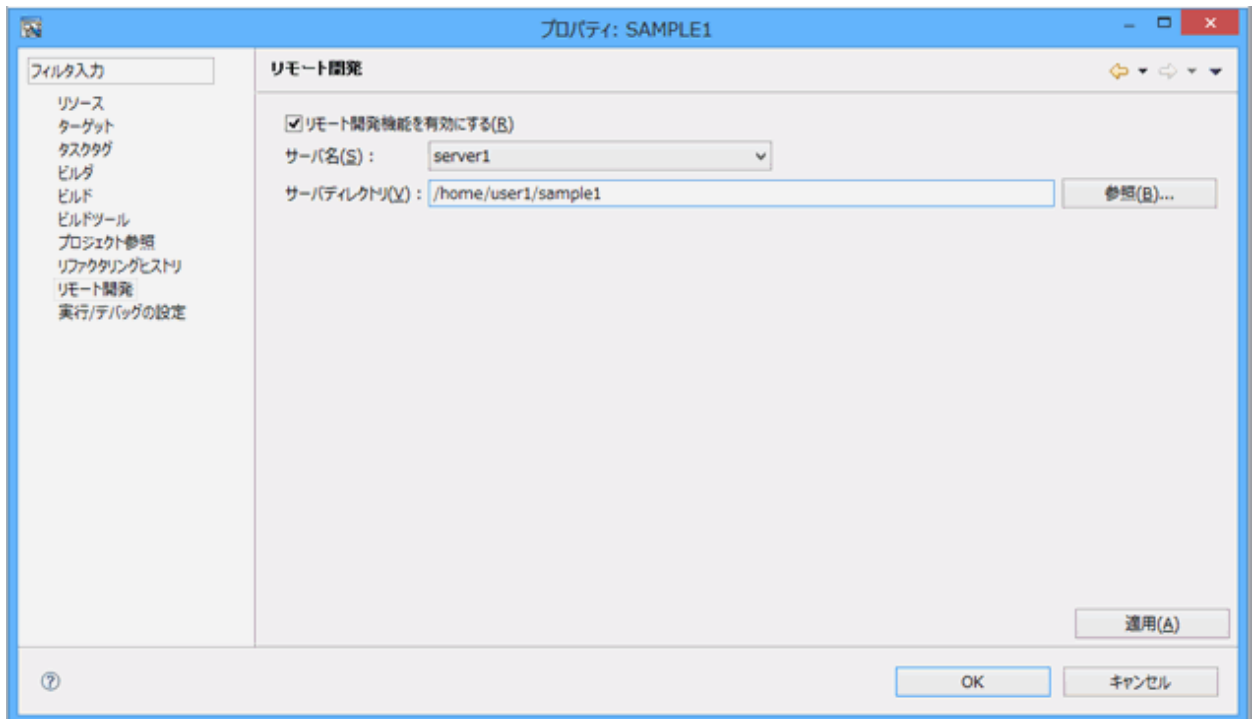
ポイント

ここで設定したサーバ情報はワークスペース間で共有されるため、他のワークスペースからでも利用することができます。

2.4.3 プロジェクトで使用するサーバ情報の設定

ローカル開発用のCOBOLプロジェクトをリモート開発用のプロジェクトとして利用するには、プロジェクトごとにサーバ情報を設定します。

1. [依存]または[構造]ビューでプロジェクトを選択し、コンテキストメニューから[プロパティ]を選択します。
→ [プロパティ]ダイアログボックスが表示されます。
2. [プロパティ]ダイアログボックスの左のペインで[リモート開発]を選択すると[リモート開発]ページが表示されます。
3. 以下のサーバ情報を設定し、[OK]ボタンをクリックします。



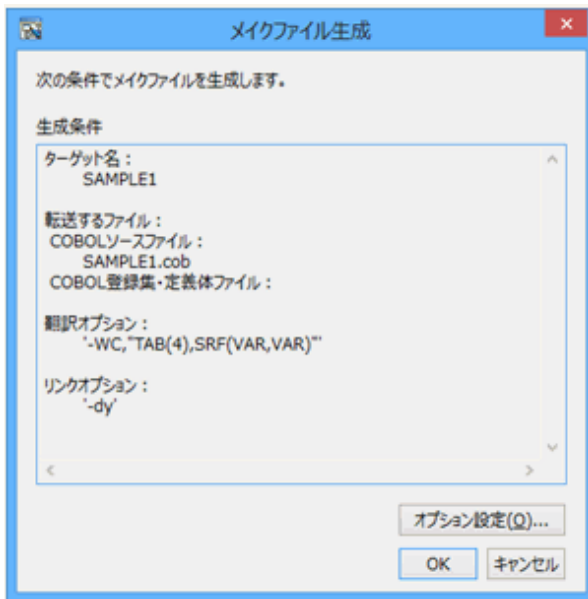
設定項目	設定内容
リモート開発機能を有効にする	選択します。
サーバ名	"サーバ情報の設定"で設定したサーバ名を選択します。
サーバディレクトリ	リモート開発で使用する開発資産の保存先ディレクトリをフルパス名で指定します。 [参照]ボタンを選択してサーバのディレクトリを参照することができます。 メイクファイル生成機能およびリモートビルド機能は、このディレクトリをカレントディレクトリとして処理を実行します。

2.4.4 メイクファイルの生成

サーバでCOBOLプログラムをビルドするためのメイクファイルを生成します。

1. メニューバーから[プロジェクト] > [リモート開発] > [メイクファイル生成]を選択します。
→ [メイクファイル生成]ダイアログボックスが表示されます。

2. ここでは何も設定せずに[OK]ボタンをクリックします。

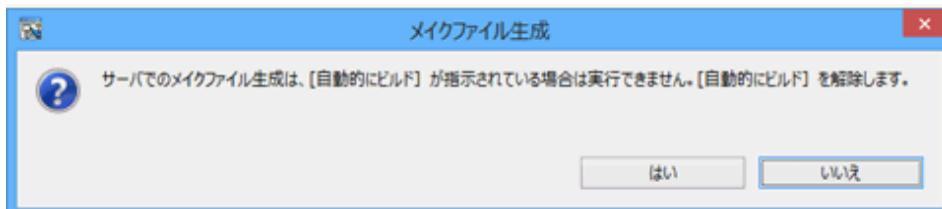


→ メイクファイルを生成するために必要となる資産がサーバへ転送され、メイクファイルが生成されます。

生成したメイクファイルは、[依存]ビューまたは[構造]ビューの[その他のファイル]フォルダーにファイル名"Makefile"で登録されます。メイクファイルの内容を確認したい場合は、メイクファイルを選択してコンテキストメニューから[開く]を選択してください。

注意

次のような確認メッセージが表示される場合、[はい]ボタンをクリックして[自動的にビルド]を解除してください。



ポイント

メイクファイル生成時のサーバでの実行結果は、[コンソール]ビューのツールバーのアイコン([コンソールを開く])から[COBOLリモート]を選択することにより確認できます。

2.4.5 サーバ側でビルド

メニューバーから[プロジェクト] > [リモート開発] > [ビルド]を選択します。

→ サーバでCOBOLプログラムがビルドされます。

ポイント

- ・ 翻訳エラーは[問題]ビューに表示されます。[問題]ビューで翻訳エラーを選択し、コンテキストメニューから[ジャンプ]を選択すると、COBOLエディタでCOBOLソースプログラムが開かれて翻訳エラー箇所がカレント行となります。

- ・ サーバでのビルド結果は、[コンソール]ビューのツールバーのアイコン([コンソールを開く])から[COBOLリモート]を選択することにより確認できます。

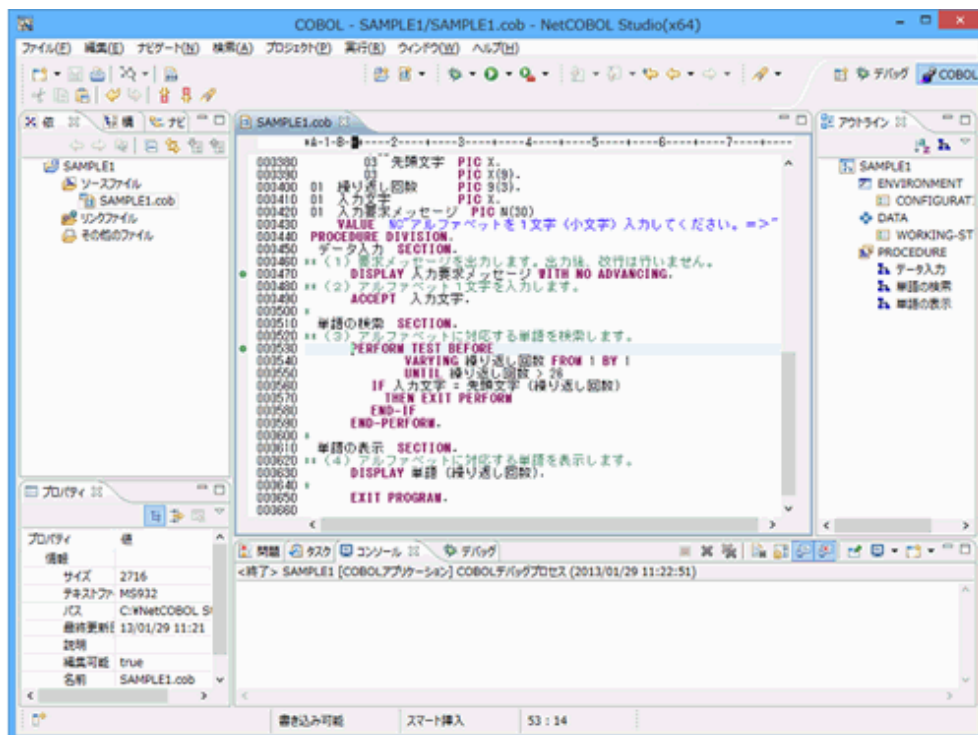
2.4.6 サーバ側でデバッグ

サーバで動作するCOBOLプログラムは、リモートデバッグを起動してデバッグします。

リモートデバッグは以下の手順で開始します。

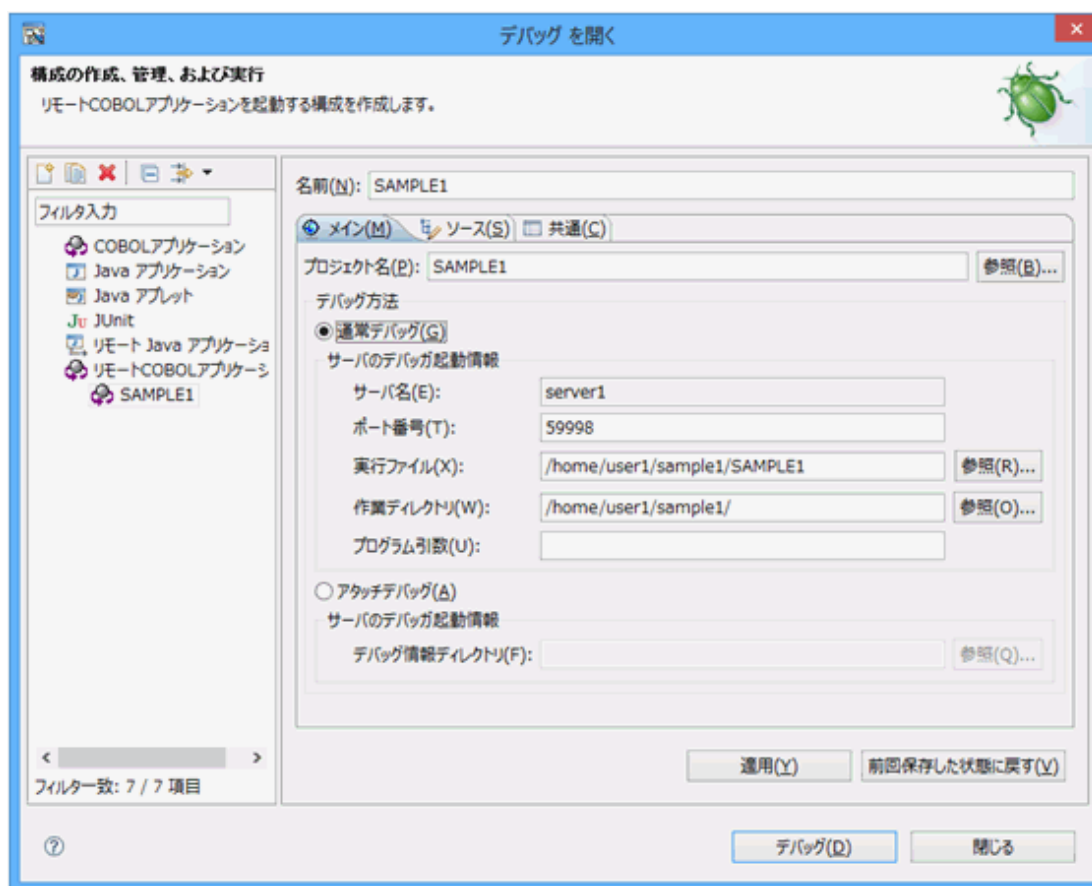
1. サーバでsvdrdsコマンドを実行し、リモートデバッグコネクタを起動します。
2. デバッグを起動する前にブレークポイントを設定します。以下の手順でブレークポイントを設定します。
 - a. COBOLエディタでブレークポイントを設定する行の垂直方向ルーラ(行番号領域の左側)にマウスカーソルを位置付けます。
 - b. マウスの左ボタンをダブルクリックします。

→ 垂直方向ルーラにブレークポイントの設定を表す  が表示されます。

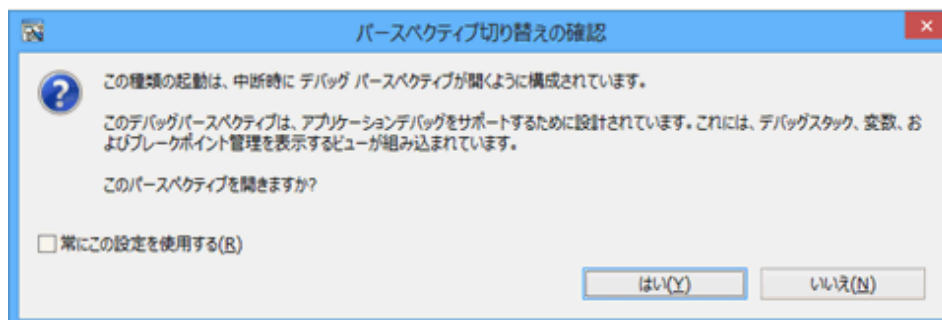


3. ブレークポイントの設定が完了したらデバッグを起動します。デバッグは以下の手順で起動します。
 - a. メニューバーから[実行] > [デバッグの構成]を選択します。
→ [デバッグを開く]ダイアログボックスが表示されます。
 - b. 左ペインで[リモートCOBOLアプリケーション]をダブルクリックして、デバッグの起動構成を作成します。

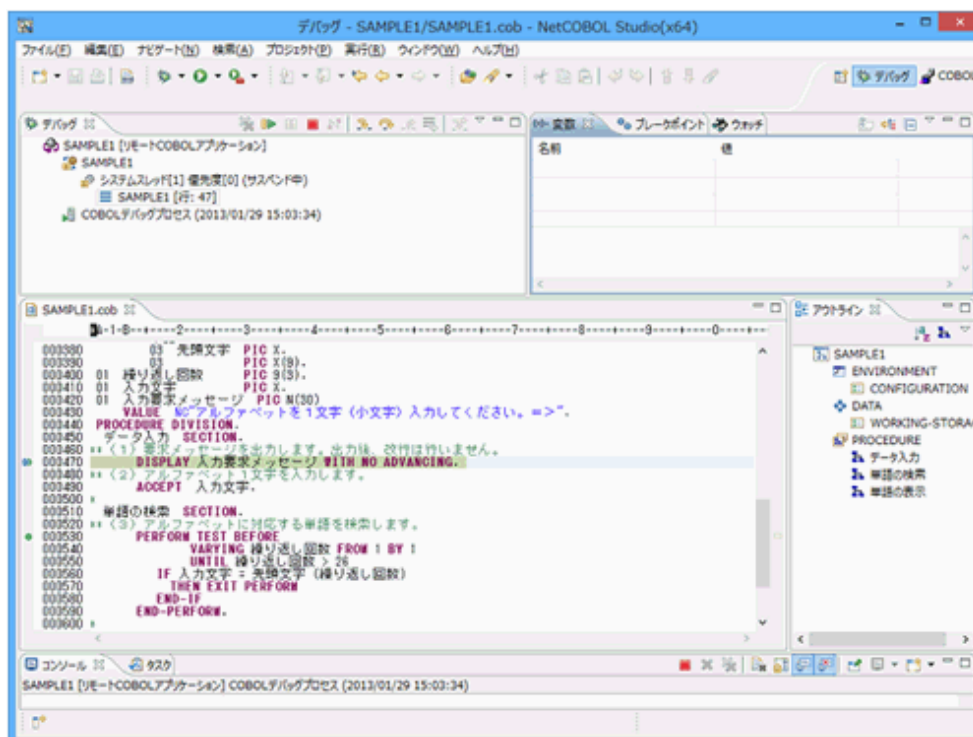
- c. [メイン]タブの[デバッグ方法]で[通常デバッグ]を選択し、[デバッグ]ボタンをクリックします。



4. 処理が最初のブレークポイントに達すると、パースペクティブを切り替えるかの確認メッセージが表示されます。[はい]ボタンを選択します。



5. パースペクティブがデバッグパースペクティブに切り替わり、最初のブレークポイントで処理が中断します。メニューバーの[実行]から目的のメニュー項目を選択して、デバッグを実行してください。



ポイント

デバッグ完了後にCOBOLパースペクティブに戻るには、ウィンドウ右上部の  を選択し、表示されたメニューから[COBOL]を選択してください。

第3章 画面帳票アプリケーションの開発

本章では、画面帳票定義体を使用した画面帳票アプリケーションについて説明します。また、基本的な画面帳票アプリケーションを作成する方法について説明します。

3.1 概要

NetCOBOLシリーズで作成できる画面帳票アプリケーション、および本章で作成する画面帳票アプリケーションの概要について説明します。

3.1.1 画面帳票アプリケーションの概要

NetCOBOLシリーズでは、「画面定義体」と呼ばれる画面フォーマット、「帳票定義体」と呼ばれる帳票フォーマットおよび帳票定義体に重ねて使用する「オーバーレイ定義体」を用いることにより、COBOLによるきめ細かい画面帳票アプリケーションを作成することができます。

画面定義体および帳票定義体には、項目の位置、項目の種別(どのような種類のデータを扱う項目か、固定的な項目かなど)、項目の属性(文字の大きさ、色など)、罫線や網がけといった装飾などを定義します。また、オーバーレイ定義体には固定的な文字や図形などを定義します。

画面定義体、帳票定義体およびオーバーレイ定義体は、COBOLプログラムから独立しているため、作成や変更が容易です。なお、画面定義体と帳票定義体を総称して、「画面帳票定義体」と呼びます。

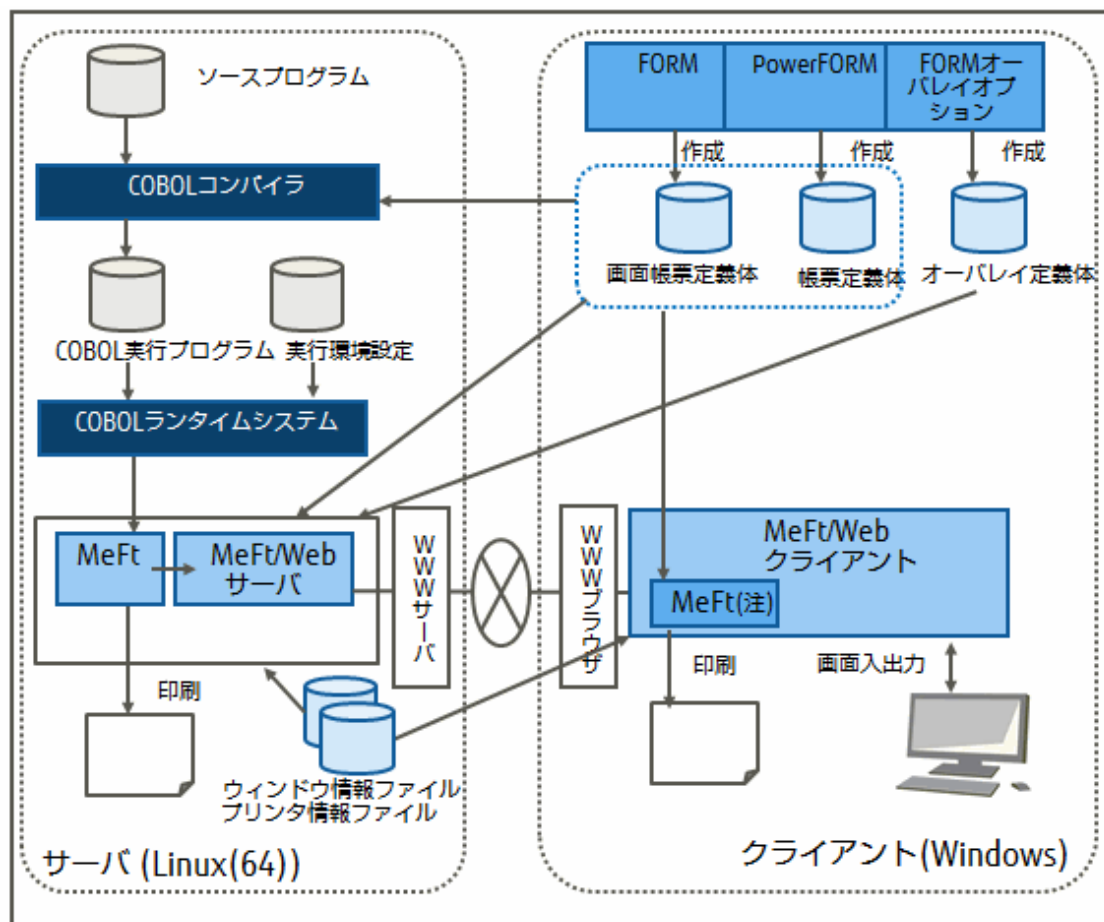
画面帳票定義体およびオーバーレイ定義体を使用したプログラムの開発および実行には、NetCOBOLのほかに次のツールも使用します。

名称	説明
FORM	画面帳票定義体を画面イメージで設計するツール。
FORMオーバーレイオプション	オーバーレイ定義体を画面イメージで設計するFORMのオプション製品。
PowerFORM	帳票定義体を画面イメージで作成する帳票設計ツール。FORMに含まれます。
MeFt	画面帳票定義体およびオーバーレイ定義体を元に帳票印刷を行うライブラリ。
MeFt/Web	Webブラウザを使って、Web サーバ上で動作する利用者プログラムをディスプレイ装置やプリンター装置に入出力することができる通信プログラム。

FORM、FORMオーバーレイオプションおよびPowerFORMは、Linux(64)版 NetCOBOLシリーズでは提供していません。Windows版をお使いください。

MeFtおよびMeFt/WebはNetCOBOLシリーズのStandard Editionおよび Enterprise Editionに含まれています。

COBOL、FORM、FORMオーバーレイオプション、PowerFORM、MeFtおよびMeFt/Webの関連図を示します。



注: MeFt/Webサーバからダウンロードされます。

COBOLプログラムからFORM、MeFt/Webを使用して画面入出力を行う場合、表示ファイルによるアプリケーションを作成します。表示ファイルによる画面帳票入出力では、通常のファイルを扱うのと同じようにWRITE文やREAD文を使用します。つまり、WRITE文で画面やプリンターへの出力を行い、READ文で画面から入力します。

プログラムは、画面およびプリンターとのデータの受渡し手段としてレコードを使用します。画面帳票定義体に定義されたデータ項目のレコードは、COBOLのCOPY文を使って、翻訳時にプログラムに取り込むことができます。そのため、画面帳票の入出力のためのレコードの定義をCOBOLプログラムに記述する必要はありません。

なお、データ項目のウィンドウ内での位置や印刷位置など、ウィンドウやプリンターの制御はMeFt(またはMeFt/Web)が行うため、COBOLプログラムでは意識する必要はありません。

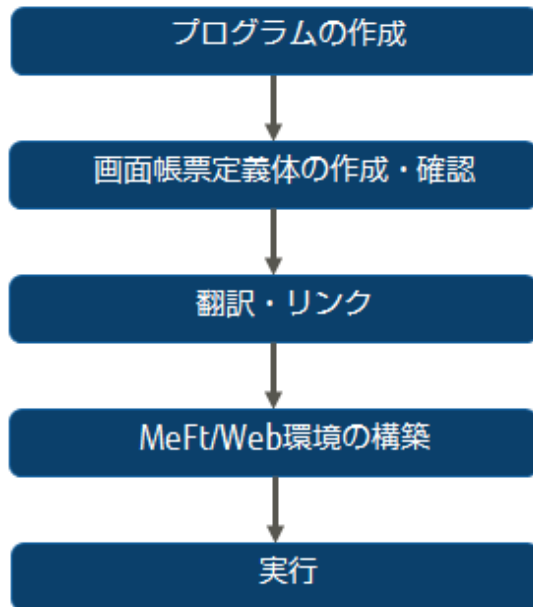
3.1.2 作成するアプリケーションについて

この章で作成するアプリケーションでは、1つの画面定義体と1つの帳票定義体を使用します。画面定義体を使用してデータを入力し、入力されたデータは帳票定義体を使用して印刷します。

なお、ここで作成するアプリケーションはサンプルプログラム(“6.3 表示ファイル機能を使ったプログラム(sample03)”)として提供しています。

3.1.3 アプリケーション開発の流れ

画面帳票定義体を使用した画面帳票アプリケーションの開発の流れを次に示します。



3.2 表示ファイルのプログラミング

表示ファイル機能を使って画面入出力を行うときのプログラム記述について、COBOLの各部ごとに説明します。

3.2.1 環境部(ENVIRONMENT DIVISION)

表示ファイルを定義します。表示ファイルは、通常のファイルを定義するときと同様に、入出力節のファイル管理段落にファイル管理記述項を記述します。

以下に、COBOLプログラムの記述例とファイル管理記述項に指定できる内容を示します。

COBOLプログラムの記述例

[sample3.cob]

表示ファイル(画面機能)の定義

```
ENVIRONMENT DIVISION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
    SELECT 画面の表示先 ASSIGN TO GS-DSPFILE  
        SYMBOLIC DESTINATION IS "DSP"  
    FORMAT          IS 画面の名前  
    GROUP           IS 項目群の名前  
    SELECTED FUNCTION IS アテンションの種類  
    PROCESSING MODE  IS 処理種別  
    FILE STATUS     IS 画面の状態.
```

[denpyou.cob]

表示ファイル(帳票機能)の定義

```
ENVIRONMENT DIVISION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
    SELECT 帳票の印刷先 ASSIGN TO GS-PRTFILE  
        SYMBOLIC DESTINATION IS "PRT"  
    FORMAT          IS 帳票の名前  
    GROUP           IS 項目群の名前  
    PROCESSING MODE  IS 処理種別
```

ファイル管理記述項に指定できる内容

指定場所	情報の種類	指定する内容	必須／任意
SELECT句	ファイル名	COBOL プログラム中で使用するファイル名を指定します。	必須
ASSIGN句	ファイル参照子	「GS-ファイル識別名」の形式で指定します。このファイル識別名は、実行時に使用するウィンドウ情報ファイルまたはプリンタ情報ファイルのファイル名を設定する環境変数情報になります。	必須
SYMBOLIC DESTINATION句	あて先種別	データの入出力のあて先を指定します。画面機能では「DSP」(または省略)、帳票機能では「PRT」を指定します。	任意(画面) 必須(帳票)
FORMAT句	データ名	作業場所節または連絡節で、8桁の英数字項目として定義したデータ名を指定します。このデータ名には、画面帳票定義体名を設定します。	必須
GROUP 句	データ名	作業場所節または連絡節で、8桁の英数字項目として定義したデータ名を指定します。このデータ名には、入出力の対象となる項目群名を設定します。	必須
PROCESSING MODE句	データ名	作業場所節または連絡節で、2桁の英数字項目として定義したデータ名を指定します。このデータ名には、入出力の処理種別を設定します。	任意
UNIT CONTROL句	データ名	作業場所節または連絡節で、6桁の英数字項目として定義したデータ名を指定します。このデータ名には、制御情報を設定します。	任意
SELECTED FUNCTION句	データ名	作業場所節または連絡節で、4桁の英数字項目として定義したデータ名を指定します。このデータ名には、READ文完了時にアテンション情報が通知されます。画面機能で指定します。	任意
FILE STATUS句	データ名	作業場所節または連絡節で、2桁および4桁の英数字項目として定義したデータ名を指定します。このデータ名には、入出力処理の実行結果が設定されます。なお、4桁のデータ名の領域には、実行結果の詳細情報が設定されます。	任意

注) 詳細は、“NetCOBOLユーザズガイド”の「8.5.4 プログラムの記述」(帳票機能の場合)および「9.1.4 プログラムの記述」(画面機能の場合)をご参照ください。

3.2.2 データ部(DATA DIVISION)

データ部には、表示ファイルのレコード定義およびファイル管理記述項に指定したデータの定義を記述します。表示ファイルのレコードは、XMDLIBを指定したCOPY文を使って画面帳票定義体から取り込むことができます。

以下にCOBOLプログラムの記述例を示します。

[sample3.cob]

DATA DIVISION.			
FILE SECTION.			
FD 画面の表示先.		*>--+表示ファイルのレコードを画面帳票定義体からCOPY文で取り込み	
COPY DENPY001 OF XMDLIB.		*>--+	
WORKING-STORAGE SECTION.			
77 画面の名前	PIC X(08)	VALUE "DENPY001".	*>--+
77 項目群の名前	PIC X(08).		*> FILE-CONTROLで定義した表示ファイル
77 アテンションの種類	PIC X(04).		*> (画面機能)の各データ項目の定義
88 P F 1 キー		VALUE "F001".	*>
88 P F 2 キー		VALUE "F002".	*>

88	P F 3 キー	VALUE "F003".	*>
77	処理種別	PIC XX.	*>
77	画面の状態	PIC XX.	*>
88	画面のアクセス成功	VALUE "00" THRU "04".	*>--+

[denpyou.cob]

```

DATA DIVISION.
FILE SECTION.
  FD  帳票の印刷先.                *>--+ 表示ファイルのレコードを帳票定義体からCOPY文で取り込み
    COPY  DENPY002 OF XMDLIB.      *>--+
WORKING-STORAGE SECTION.
  77  帳票の名前                    PIC X(8)  VALUE "DENPY002".  *>--+
  77  項目群の名前                  PIC X(8)  VALUE "GRP001 ".  *> | FILE-CONTROLで定義した表示ファイル
  77  処理種別                      PIC XX.    *> | (帳票機能) の各データ項目の定義
  77  詳細情報                      PIC X(4).  *> |
  77  印刷の状態                    PIC XX.    *>--+

```

3.2.3 手続き部(PROCEDURE DIVISION)

画面入出力および帳票出力の開始にはOPEN文を、終了にはCLOSE文を使用します。また、画面および帳票の入出力には、通常のファイル処理を行うときと同様に、READ文およびWRITE文を使用します。

手続き部について、画面機能と帳票機能に分けて説明します。

3.2.3.1 画面機能

COBOLプログラムの記述例

[sample3.cob]

```

      OPEN I-0 画面の表示先.
:
      画面を表示する 1.
      MOVE "GRP001" TO 項目群の名前.
      MOVE " "      TO 処理種別.
      WRITE DENPY001.
:
      画面からデータを読み込む 1.
      READ 画面の表示先.
      IF P F 3 キー
      :
      IF P F 2 キー
      :
      IF NOT 画面のアクセス成功
      ::
      画面を表示する 2.
      MOVE "GRP001" TO 項目群の名前.
      MOVE "CL"     TO 処理種別.
      WRITE DENPY001.
:
      画面からデータを読み込む 2.
      MOVE "NE"     TO 処理種別.
      READ 画面の表示先.
      IF P F 3 キー
      :
      IF P F 2 キー
      :
      IF NOT 画面のアクセス成功
      :
:
      CLOSE 画面の表示先.

```

画面機能のREAD文およびWRITE文について

画面を表示するときには表示ファイルのレコード名を指定したWRITE文を、画面からデータを読み込むときには表示ファイルを指定したREAD文を使います。

WRITE文を実行する前には、画面出力に使用する画面定義体の名前をFORMAT句に指定したデータ名に設定し、出力の対象となる画面定義体の項目群名をGROUP句に指定したデータ名に設定します。

また、画面出力の処理種別(モード)をPROCESSING MODE句に指定したデータ名に設定します。

入力の対象となる画面定義体の項目群名と入力の処理種別(モード)が出力時と異なるときは、READ文を実行する前に、画面定義体の項目群名をGROUP句に指定したデータ名に設定し、処理種別(モード)をPROCESSING MODE句に指定したデータ名に設定します。

画面定義入力中にファンクションキーなどが押されると、READ文が完了してCOBOLプログラムに入力結果の情報が通知されます。そのとき、SELECTED FUNCTION句に指定したデータ名にアテンション情報が通知されます。

3.2.3.2 帳票機能

COBOLプログラムの記述例

```
OPEN OUTPUT 帳票の印刷先.
:
MOVE "PW"      TO 処理種別.
MOVE "P001"    TO 詳細情報.
WRITE DENPY002.
:
CLOSE 帳票の印刷先.
```

帳票機能のWRITE文について

帳票を出力するときには、表示ファイルのレコード名を指定したWRITE文を実行します。

WRITE文を実行する前には、印刷に使用する帳票定義体の名前をFORMAT句に指定したデータ名に設定し、印刷の対象となる帳票定義体の項目群名をGROUP句に指定したデータ名に設定します。

また、帳票出力の処理種別(モード)をPROCESSING MODE句に指定したデータ名に設定します。

3.2.4 エラー処理

表示ファイルの各命令(OPEN文、WRITE文、READ文、CLOSE文)の実行結果は、FILE STATUS句に指定したデータ名に通知されます。FILE STATUS句に指定したデータ名のうち2桁のデータ名の領域には、成功時は「00」、「04」が通知され、不成功時は「90」、「99」、「9E」のいずれかが通知されます。また、4桁のデータ名の領域には、詳細結果として上記の4種類のエラーにMeFtの通知コードが付加されたものが通知されます。例えば、MeFtの通知コードが「22」であった場合、FILE STATUS句に指定した4桁のデータ領域には「9022」が通知されます。

入出力文の後に、このデータ名の内容をチェックする文を記述することによって、プログラムで入出力文の結果に応じた処理手続きを実行することができます。

FILE STATUS句の使用例を次に示します。

```
SELECT 画面の表示先 ASSIGN TO GS-DSPFILE
:
FILE STATUS      IS 画面の状態 1 画面の状態 2.
:
WORKING-STORAGE SECTION.
77 画面の状態 1      PIC X(02).
88 画面のアクセス成功 VALUE "00" THRU "04".
77 画面の状態 2      PIC X(04).
:
PROCEDURE DIVISION.
OPEN I-O 画面の表示先.
```

```
WRITE DENPYO01.  
IF NOT 画面のアクセス成功  
THEN GO TO 画面表示失敗処理.
```

3.3 画面帳票定義体の作成

画面帳票定義体の作成について、基本的な操作を説明します。操作の詳細な説明については、FORMのマニュアルをご参照ください。
なお、ここで作成する画面帳票定義体(DENPYO01.smd)は、sample03で提供しています。



注意

FORM、FORMオーバレイオプションおよびPowerFORMは、Linux(64)版 NetCOBOLシリーズでは提供していません。Windows版をお使いください。

3.3.1 画面定義体の作成

ここでは、次に示すような画面定義体を作成する方法を説明します。

次に示す手順で画面定義体を作成していきます。

1. 画面定義体の属性設定

画面定義体の定義体サイズ(縦幅／横幅)など、画面定義体全体に対する属性を設定します。

2. 項目の配置と属性、レコード情報、罫線情報の設定

データを入力する項目の配置など、画面定義体のレイアウトを作成します。また、配置した項目に対する属性設定、レコード定義および罫線情報を設定します。

3. アテンション情報の設定

キーボードで押されたファンクションキーをプログラムに通知する情報(アテンション情報)を設定します。

3.3.1.1 FORMの起動

FORMの編集画面が表示されます。



3.3.1.2 画面定義体の属性設定

1. [ファイル]メニューから、[新規作成] > [画面定義体]を選択します。
2. [ファイル]メニューから、[プロパティ] > [画面帳票定義体]を選択すると、画面定義体のプロパティ画面が表示されます。
3. プロパティの画面で、以下に示す内容を設定します。

設定箇所	設定内容
定義体名	伝票画面
縦幅(行)	18
横幅(桁)	80

4. [適用]ボタンをクリックすると、編集画面に戻ります。

3.3.1.3 項目の配置と属性、レコード情報、罫線情報の設定

項目を配置し、その属性を設定します。

項目を配置するには、FORMの[項目]メニューから配置したい項目の種類を選択し、表示されるプロパティの画面で属性を設定します。

ここでは、例として、見出しの「コード」という文字列の項目を配置し、属性を設定する方法を説明します。

1. 編集画面で、見出しを配置したい位置にカーソルを移動します。

2. 見出しのように表示する文字が固定されている項目を「固定リテラル項目」といいます。[項目]メニューから「固定リテラル」を選択します。
3. 固定リテラル項目のプロパティ画面が表示されますので、次に示す内容を設定します。

設定箇所	設定内容
文字列	コード
日本語編集の「拡大」	標準

4. [適用]ボタンを選択すると、プロパティ画面が終了し、見出しの項目が編集画面上に配置されます。

5. その他の項目は、次に示す内容で設定します。

「項目」メニューでの項目の種類	項目名	文字列	位置 (行、桁)	種別	横幅 (文字数)	桁数	編集形式
日本語	メッセージ		1行、5桁	出力	15		赤、メッセージ表示
数字	年		3行、55桁	出力		4	ZZZZZZ9
固定リテラル		.	3行、60桁				
数字	月		3行、62桁	出力		2	ZZZZZZ9
固定リテラル		.	3行、65桁				
数字	日		3行、67桁	出力		2	ZZZZZZ9
固定リテラル		(	3行、70桁				
日本語	曜日		3行、73桁	出力	1		
固定リテラル		)	3行、76桁				
英数字	商品コード		7行、8桁	入出力	4		
固定リテラル		商品名	5行、29桁				
日本語	商品名		7行、15桁	出力	20		
固定リテラル		数量	5行、58桁				
数字	数量		7行、58桁	入出力		4	ZZZZZZ9
固定リテラル		単価	5行、65桁				
数字	単価		7行、65桁	出力		4	ZZZZZZ9
固定リテラル		金額	5行、72桁				
数字	金額		7行、72桁	出力		5	ZZZZZZ9
固定リテラル		合計	17行、61桁				
数字	合計金額		17行、71桁	出力		6	ZZZZZZ9
固定リテラル		PF1:計算	17行、8桁				青
固定リテラル		PF2:次の伝票を印刷	17行、24桁				青
固定リテラル		PF3:終了	18行、8桁				赤
固定リテラル		PF4:印刷	18行、24桁				青

全ての項目の配置、属性設定が終了すると、以下のような状態になります。

FORM [画面: .SMD]

ファイル(F) 編集(E) 表示(V) 形式(T) 項目(I) ヘルプ(H) 図形(S) オプション(O) ヘルプ(H)

編集画面 新規 自由

1 NNNNNNNNNNNNNNNNNNN

2

3 ZZZ9 . Z9 . Z9 (N)

4

5 コード 商品名 数量 単価 金額

6 XXXX NNNNNNNNNNNNNNNNNNNNNNN ZZZ9 ZZZ9 ZZZ9

7

8

9

10

11

12

13

14

15

16

17 PF1: 計算 PF2: 次の伝票を印刷 合計 ZZZZZ9

18 PF3: 終了 PF4: 印刷

100% 1 1

6. [表示]メニューから[レコード定義へ]を選択します。
7. [オプション]メニューから[基本レコードを生成]を選択すると、編集画面上に定義した項目のレコード定義が生成されます。

FORM [画面: .SMD]

ファイル(F) 編集(E) 表示(V) 形式(T) 項目(I) ヘルプ(H) 図形(S) オプション(O) ヘルプ(H)

レコード定義 新規 自由

番号	ヘル	項目名	英数字形	項目名式	桁数	位置	桁	繰返数	句	横幅	縦幅	固定	小数	形式
1	01	メッセージ	R001	N	15	1	6							
2	01	年	R002	Z	4	3	55							
3	01	月	R003	Z	2	3	62							
4	01	日	R004	Z	2	3	67							
5	01	曜日	R005	N	1	3	73							
6	01	商品コード	R006	X	4	7	7							
7	01	商品名	R007	N	20	7	15							
8	01	数量	R008	Z	4	7	58							
9	01	単価	R009	Z	4	7	65							
10	01	金額	R00A	Z	5	7	72							
11	01	金額合計	R00B	Z	6	17	71							

*** **

8. 上記のレコード定義画面で、「商品コード」から「金額」までの項目を選択した状態で、[項目]メニューから[集団項目定義]を選択します。

集団項目が作成され、「商品コード」から「金額」までの項目のレベルが「02」になります。

番号	レベル	項目名	英数字形	項目名式	桁数	位置	方向	繰返し数	繰返し方向	繰返し幅	繰返し個数	繰返し形式
1	01	メッセージ	R001	N	15	1	0					
2	01	年	R002	Z	4	3	55					
3	01	月	R003	Z	2	3	62					
4	01	日	R004	Z	2	3	67					
5	01	曜日	R005	N	1	3	73					
6	01		MAS001	G								
7	02	商品コード	R006	X	4	7	7					
8	02	商品名	R007	N	20	7	15					
9	02	数量	R008	Z	4	7	58					
10	02	単価	R009	Z	4	7	65					
11	02	金額	R00A	Z	5	7	72					
12	01	金額合計	R00B	Z	6	17	71					

9. 作成された集団項目を選択した状態で、[項目]メニューから[項目名]を選択します。[項目名]ダイアログボックスが表示されますので、項目名として「伝票」を入力します。
10. 集団項目に対して、繰返し定義を行うため、[項目]メニューから[繰返し定義]を選択します。表示されたダイアログボックスから、以下を入力します。
- 繰返し数:5
 - 方向:下
 - 横幅:0
 - 縦幅:2
 - 折返し回数:0
 - 繰返しの展開:チェック

[適用]ボタンを押すと、以下のようになります。

番号	レベル	項目名	英数字形	項目名式	桁数	位置	方向	繰返し数	繰返し方向	繰返し幅	繰返し個数	繰返し形式
1	01	年	R002	Z	4	3	55					
2	01	月	R003	Z	2	3	62					
3	01	日	R004	Z	2	3	67					
4	01	曜日	R010	N	1	3	73					
5	01	伝票	MAS001	G				5	下		2	
6	02	商品コード	R011	X	4	7	8					
7	02	商品名	R006	N	20	7	15					
8	02	数量	R007	Z	4	7	58					
9	02	単価	R008	Z	4	7	65					
10	02	金額	R009	Z	5	7	72					
11	01	金額合計	R030	Z	6	17	71					
12	01	メッセージ	R005	N	15	1	5					

11. [表示]メニューから[編集画面へ戻る]を選択すると、以下の状態になります。

The screenshot shows a window titled 'FORM [画面: SMD]' with a menu bar (File, Edit, Display, Format, Item, Group, Shape, Action, Help) and a toolbar. The main area contains a form with a table and various controls.

コード	商品名	数量	単価	金額
XXXX	NNNNNNNNNNNNNNNNNNNNNN	ZZZ9	ZZZ9	ZZZZ9
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

At the bottom, there are function keys: PF1: 計算, PF2: 次の伝票を印刷, PF3: 終了, PF4: 印刷. A '合計' (Total) row shows 'ZZZZZ9'.

12. [表示]メニューの[シンボルメニュー]をチェックすると、シンボルメニューが表示されます。
これを使用して、罫線を設定します。



13. [表示]メニューの[シンボルメニュー]をチェックすると、シンボルメニューが表示されます。
これを使用して、罫線を設定します。

The screenshot shows the FORM editor window titled "FORM [画面: DENPYO01.SMD]". The menu bar includes: ファイル(F), 編集(E), 表示(Y), 形式(I), 項目(I), データ(G), 図形(S), マクロ(Q), ヘルプ(H). The toolbar has buttons for 編集画面, 更新, and 自由. The main area displays a form layout with a table. The table has columns: コード, 商品名, 数量, 単価, 金額. The first row of data shows: XXXX, NNNNNNNNNNNNNNNNNNNNNNN, ZZZ9, ZZZ9, 6777. The table is enclosed in a red dashed border. At the bottom, there are labels for PF1: 計算, PF2: 次の伝票を入力, PF3: 終了, PF4: 印刷, and a total label 合計 with the value 6777. The status bar at the bottom shows 100%, 18, and 79.

3.3.1.4 アテンション情報の設定

アテンション情報とは

アテンション情報とは、キーボードで押されたファンクションキーをプログラムに通知する情報です。

画面定義体では、各ファンクションキーに対応するアテンション情報を設定します。プログラムでは、通知されるアテンション情報を参照すると、押されたファンクションキーを知ることができるため、アテンション情報によって処理を分けます。

利用者が設定するアテンション情報には、標準アテンション情報と拡張アテンション情報がありますが、ここでは、標準アテンション情報の設定方法を説明します。

標準アテンション情報の設定

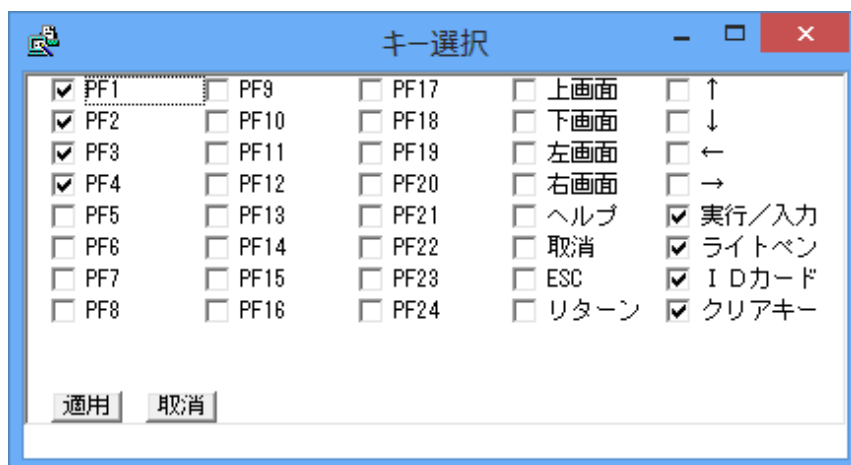
1. FORMの[形式]メニューから「標準アテンション情報」を選択し、標準アテンション情報の設定画面を表示します。

The screenshot shows the "標準アテンション情報" (Standard Attention Information) dialog box. It has a table with columns: キー名 (Key Name), データ (Data), アテンション名 (Attention Name), and 項目リテラル (Item Literal). The table contains the following data:

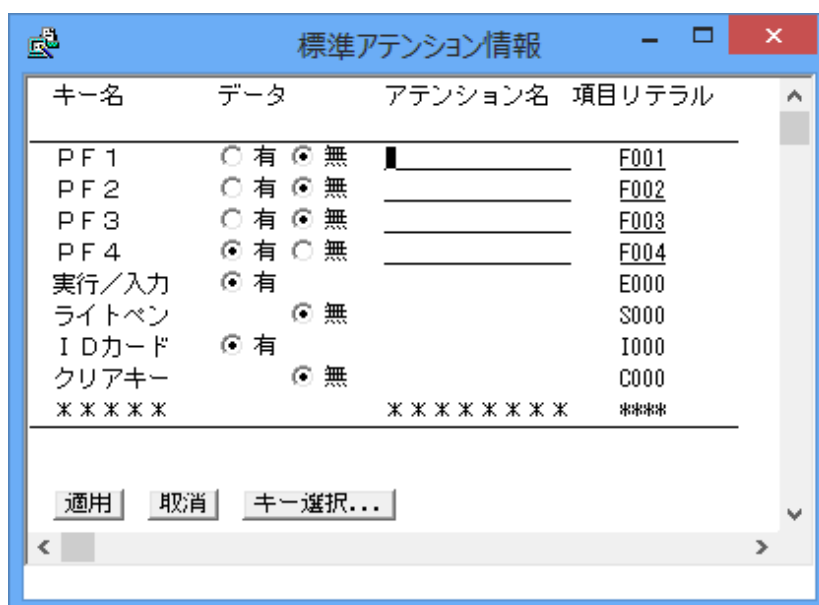
キー名	データ	アテンション名	項目リテラル
実行/入力	☒ 有		E000
ライトペン	☐ 無		S000
IDカード	☒ 有		I000
クリアキー	☐ 無		C000
*****		*****	*****

At the bottom of the dialog, there are buttons for 適用 (Apply), 取消 (Cancel), and キー選択... (Select Key...).

- 標準アテンション情報の設定画面の[キー選択]ボタンを選択し、キー選択の画面を表示します。
- キー選択の設定画面で、「PF1」～「PF4」をチェックします。



- [適用]ボタンを選択すると、キー選択の設定画面でチェックしたキーが、標準アテンション情報の設定画面に表示されます



- 標準アテンション情報の設定画面では次に示す内容を設定し、[適用]ボタンを選択します。

キー名	データ	項目リテラル
PF1	有	F001
PF2	無	F002
PF3	無	F003
PF4	有	F004

キー名	データ	アテンション名	項目リテラル
PF 1	☒ 有 ☐ 無	計算	F001
PF 2	☐ 有 ☒ 無	続行	F002
PF 3	☐ 有 ☒ 無	終了	F003
PF 4	☒ 有 ☐ 無	印刷	F004
実行/入力	☒ 有		E000
ライトペン	☒ 無		S000
IDカード	☒ 有		I000
クリアキー	☒ 無		C000
*****		*****	****

データの有無について

ファンクションキーが押されたとき、画面に入力されたデータを有効にするか、無効にするかを指定します。

「有」の場合、入力されたデータを有効とし、押されたファンクションキーのアテンション情報と共にプログラムに通知します。

「無」の場合、押されたファンクションキーのアテンション情報だけをプログラムに通知し、入力データがあっても通知しません。

項目リテラルについて

ファンクションキーを識別する名前です。プログラムでは、READ文のあとに通知される項目リテラルを判断し、押されたファンクションキーを判断できます。

3.3.1.5 定義の正当性の確認

定義した内容にエラーがないか、定義エラー一覧画面で確認します。

定義エラー一覧画面は、[オプション]メニューから[定義エラー一覧]を選択することで表示されます。

種別	プログラミング名	英数字名	メッセージ
			定義エラーはありません。

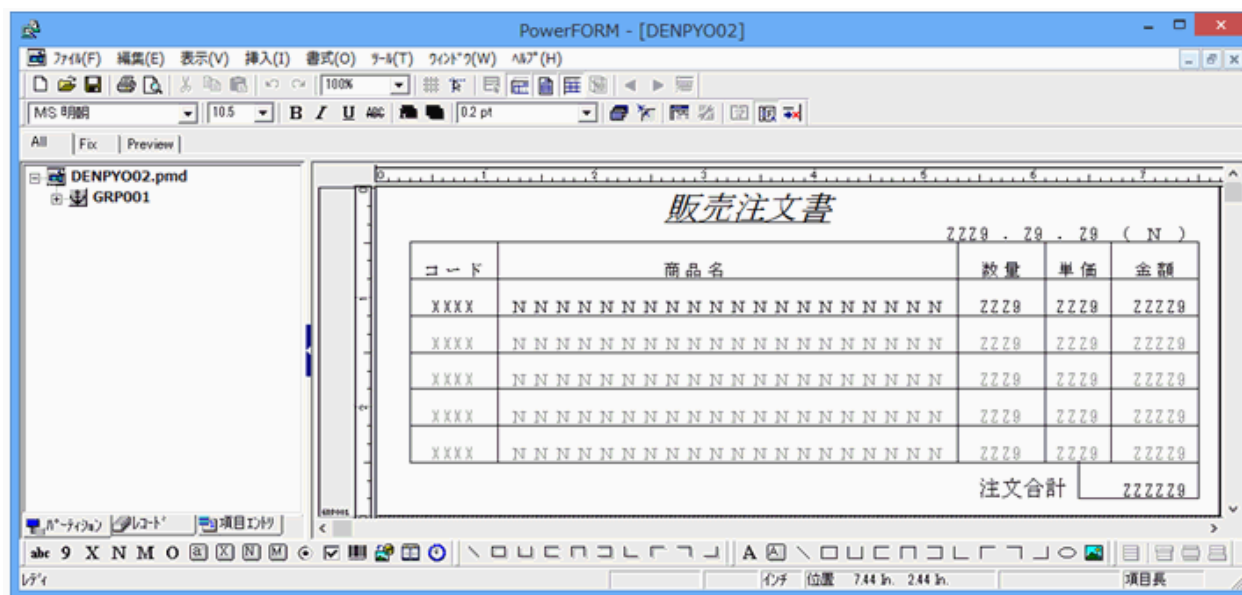
3.3.1.6 画面定義体の保存

[ファイル]メニュー→[閉じる](または[上書き保存]、[名前を付けて保存])から[画面帳票定義体]を選択します。

[名前を付けて保存]ウィンドウで、「C:\Samples」を選択し、「DENPYO01.smd」という名前を付けて画面定義体を保存します。

3.3.2 帳票定義体の作成

ここでは、PowerFORMを使用して次に示すような帳票定義体を作成する方法を説明します。



次に示す手順で帳票定義体を作成していきます。

1. 帳票定義体の属性設定

帳票定義体の印刷範囲(縦幅／横幅)など、帳票定義体全体に対する属性を設定します。

2. 項目の配置と属性設定

データを出力する項目の配置など、帳票定義体のレイアウトを作成します。また、配置した項目に対する属性を設定します。

3. 繰返しの設定

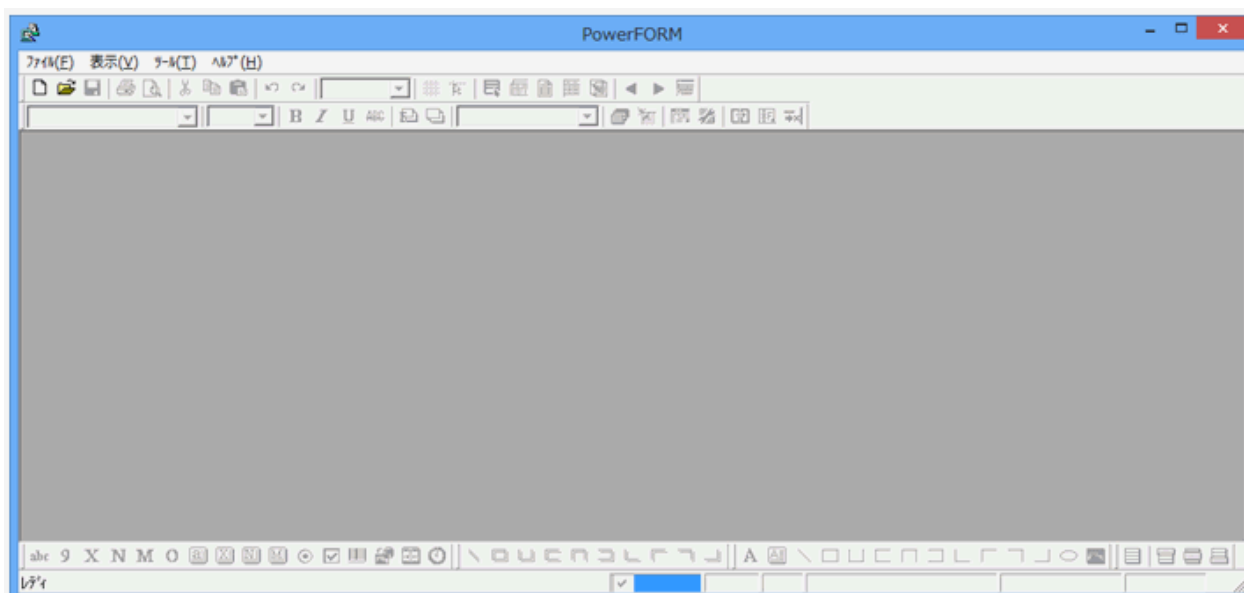
同じ属性を持つ項目を複数設定します。

4. 罫線の定義

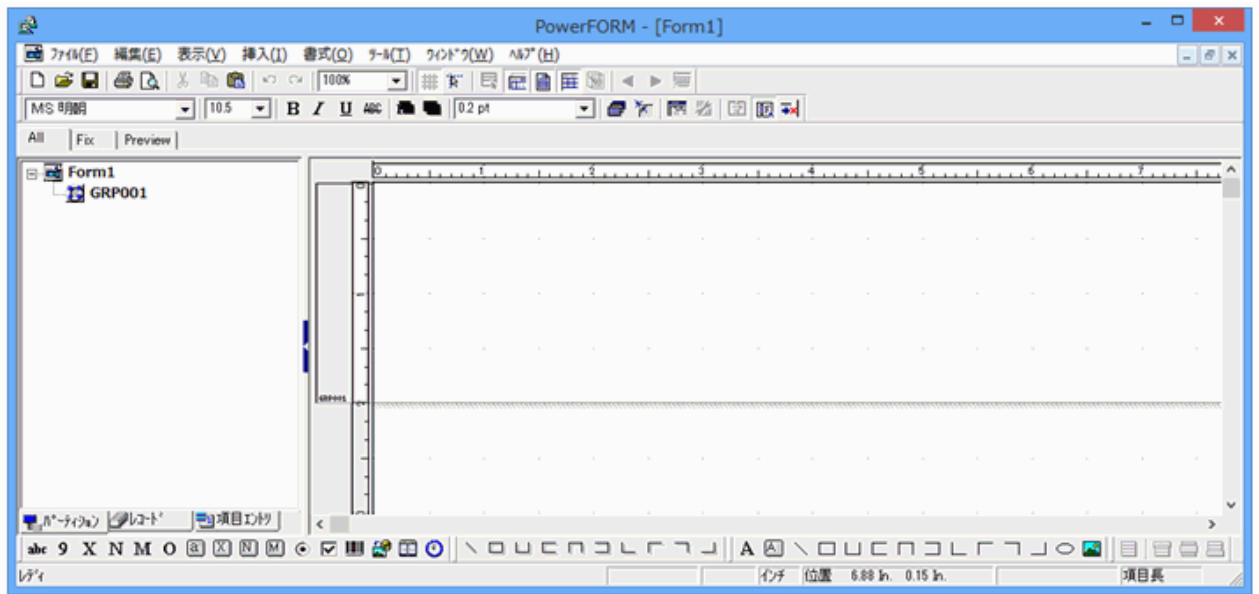
帳票定義体に罫線を定義します。

3.3.2.1 PowerFORMの起動

1. [スタート]> お使いのNetCOBOL製品名 > [PowerFORM]を選択して、PowerFORMを起動します。
PowerFORMの編集画面が表示されます。



2. PowerFORMの[ファイル]メニュー-[新規]から“新規”を選択すると、以下の編集画面が表示されます。



3.3.2.2 項目の配置と属性設定

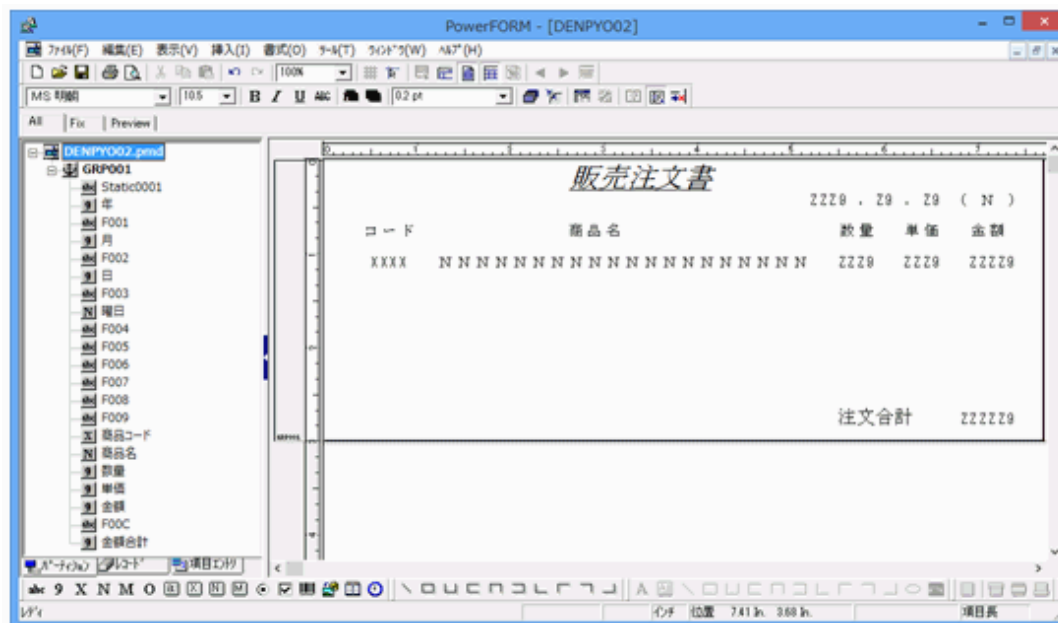
項目の配置と属性設定は、GRP001を選択して、[挿入]メニューの[項目]から行います。

各項目は、次に示す内容で設定します。

「項目」メニューでの項目の種類	項目名	文字列	項目長	位置(縦/横)	フォント	編集形式
固定リテラル	Static0001	販売注文書		0.03"/2.63"	斜体、22ポ	
数字	年		4	0.35"/5.20"		数値、ZZZZZZ9
固定リテラル	F001	.		0.35"/5.70"	標準、10.5ポ	
数字	月		2	0.35"/5.90"		数値、ZZZZZZ9
固定リテラル	F002	.		0.35"/6.20"	標準、10.5ポ	
数字	日		2	0.35"/6.40"		数値、ZZZZZZ9
固定リテラル	F003	(		0.35"/6.70"	標準、10.5ポ	
日本語	曜日		2	0.35"/7.00"		
固定リテラル	F004	)		0.35"/0.20"	標準、10.5ポ	
固定リテラル	F005	コード		0.69"/0.40"	標準、10.5ポ	
英数字	商品コード		4	1.02"/0.50"		
固定リテラル	F006	商品名		0.69"/2.60"	標準、10.5ポ	
日本語	商品名		40	1.02"/1/20"		
固定リテラル	F007	数量		0.69"/5.50"	標準、10.5ポ	
数字	数量		4	1.02"/5/50"		数値、ZZZZZZ9
固定リテラル	F008	単価		0.69"/6.20"	標準、10.5ポ	
数字	単価		4	1.02"/6.20"		数値、ZZZZZZ9
固定リテラル	F009	金額		0.69"/6.90"	標準、10.5ポ	
数字	金額		5	1.02"/6.90"		数値、ZZZZZZ9
固定リテラル	F00C	注文合計		2.64"/5.51"	標準、14ポ	

「項目」メニューでの項目の種類	項目名	文字列	項目長	位置(縦/横)	フォント	編集形式
数字	合計金額		6	2.69"/6.80"		数値、ZZZZZZ9

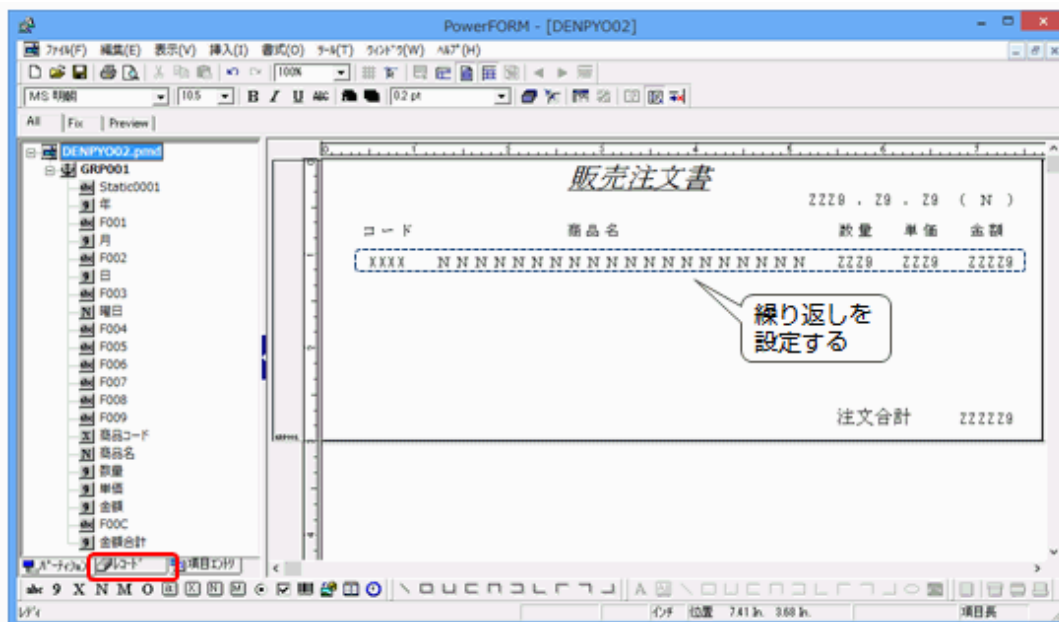
ここまでの項目の配置、属性設定が終了すると、以下の状態になります。



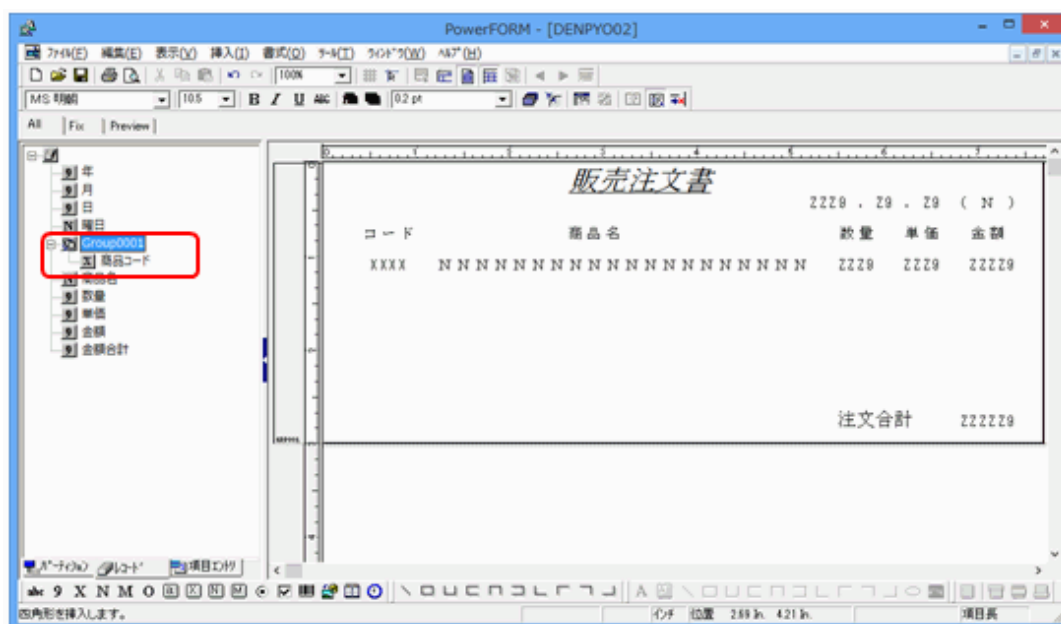
3.3.2.3 繰返しの設定

繰返しの設定とは、同じ属性を持つ項目を複数設定することをいいます。繰返しは、単一の項目にも、複数の項目から形成される集団項目にも行うことができます。ここで作成する帳票定義体では、プログラムからデータ出力を行う全ての項目を繰返し定義します。

1. 表示を[レコード]リストに切り替えます。

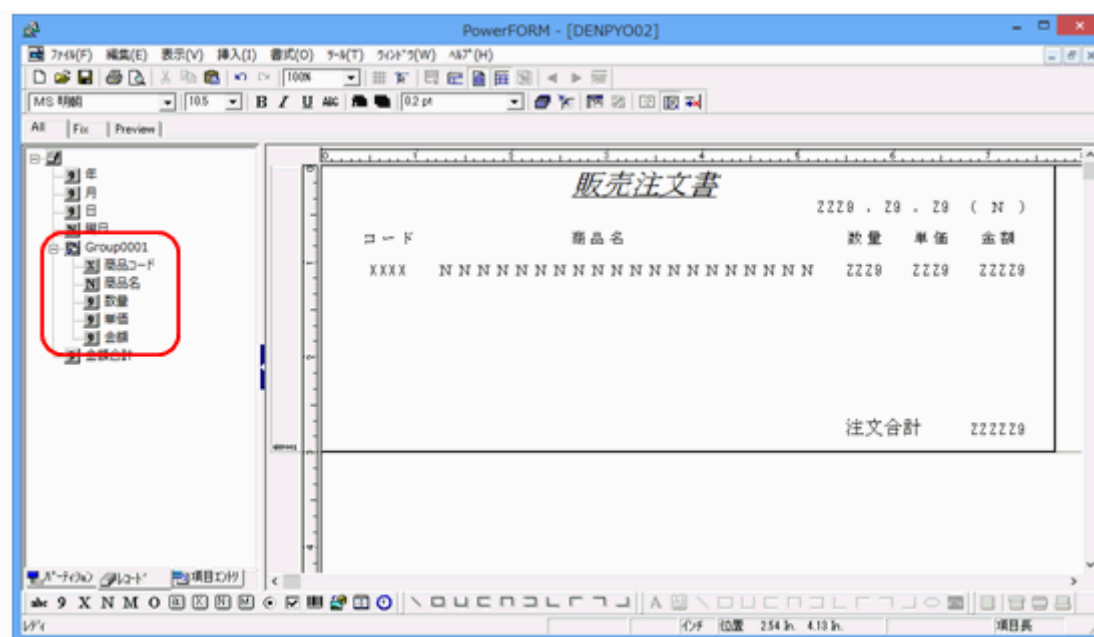


2. 「商品コード」を選択し、右クリックで表示されるコンテキストメニューから[集団項目の挿入]を選択します。



集団項目Group0001が作成されます。

3. 「商品名」を選択し、集団項目Group0001の配下にドラッグ&ドロップします。「数量」、「単価」、「金額」についても、同様に操作します。



4. Group0001を選択し、右クリックで表示されるコンテキストメニューから[プロパティ]を選択します。
5. [集団項目]ダイアログボックスで以下を設定します。

[プロパティ]タブ	項目名	伝票
[レコード情報]タブ	繰返しを行う	チェックする
	繰返し数	縦5、横1、総数5
	間隔	縦幅0.33"/横幅0.10"

ここまでの設定が終了すると、以下のような状態になります。

3.3.2.4 罫線の定義

罫線を定義する方法を説明します。

1. PowerFORM編集画面の下側にある[図形挿入]ツールバーで図形を選択します。

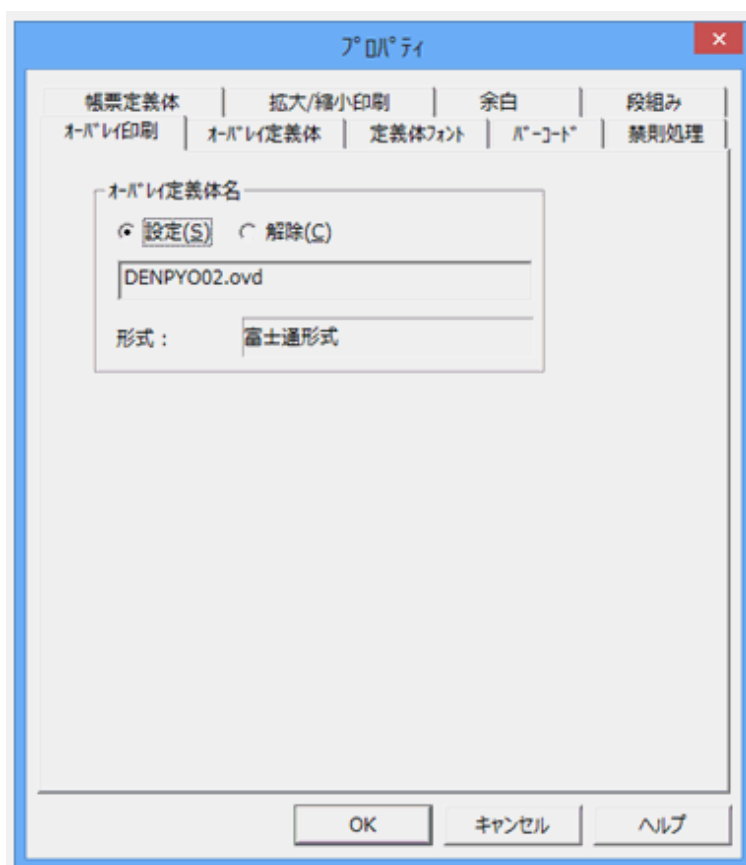


ここで作成する帳票では枠、水平線および垂直線を定義します。

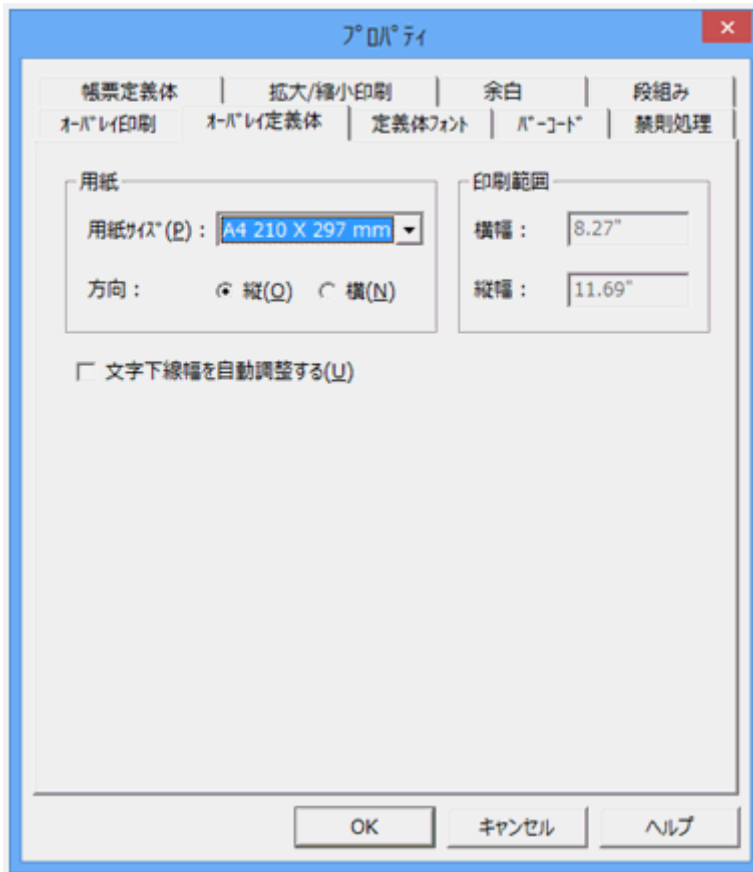
枠を定義する際は[図形挿入]ツールバーで「□」の図形を選択します。水平線はおよび垂直線は直線の図形を選択します。

2. 作図状態(マウスポインタがクロスポインタに変わります)になるので、作図範囲を開始位置から終了位置までドラッグします。
全ての罫線を作図し終わると、以下のような状態になります。

3. オーバレイ定義体名に「DENPYO02.ovd」を設定します。



なお、この例では[オーバーレイ定義体]タブの設定は特に変更しません。



3.3.2.5 帳票定義体の保存

[ファイル]メニュー→[名前を付けて保存]を選択します。

[名前を付けて保存]ウィンドウで「C:\¥Samples」を選択し、「DENPYO02.pmd」という名前を付けて帳票定義体を保存します。

3.4 プログラムの翻訳とリンク

以下の方法で、“[3.2 表示ファイルのプログラミング](#)”で作成したプログラムを翻訳・リンクします。

```
cobol -c denpyou.cob
cobol -M -osample3 sample3.cob denpyou.o
```

実行可能ファイル(sample3)が作成されます。

画面帳票定義体に関する環境変数には次のようなものがあり、必要に応じて設定します。

なお、ここで作成するアプリケーションでは、COBOLソースファイルと画面帳票定義体ファイルが同じディレクトリにあるため、これらの環境変数を設定する必要はありません。

- 環境変数SMED_SUFFIX (画面帳票定義体ファイルの拡張子)
- 環境変数FORMLIB (画面帳票定義体ファイルのディレクトリ)



注意

翻訳環境がUnicode(UTF-8)ロケールの場合、Charset Managerが必要です。以下の環境変数を設定してください。

```
LD_LIBRARY_PATH=/opt/FSUniconv/lib64
```

3.5 MeFt/Web環境の構築

表示ファイルを使用したアプリケーションの実行は、MeFt/Webを使用します。MeFt/Web環境の構築は、“[第4章 MeFt/Web環境の構築](#)”を参照してください。

3.6 実行

実行環境情報の設定

画面帳票アプリケーションを実行する場合、実行環境情報として、画面入出力用のファイルのASSIGN句で定義されたファイル識別名にウィンドウ情報ファイル名を設定し、帳票印刷用のファイルのASSIGN句で定義されたファイル識別名にプリンタ情報ファイル名を設定します。

[sample3.cob]

```
SELECT 画面の表示先 ASSIGN TO GS-DSPFILE
```

ファイル識別名 : DSPFILE

[denpyou.cob]

```
SELECT 帳票の印刷先 ASSIGN TO GS-PRTFILE
```

ファイル識別名 : PRTFILE

画面入出力の表示ファイルのファイル識別名「DSPFILE」にsample03に格納されているウィンドウ情報ファイル「mefwrc」を割り当て、帳票印刷の表示ファイルのファイル識別名「PRTFILE」とsample03に格納されているプリンタ情報ファイル「mefprc」を割り当てます。

[sample3.sh](起動用のシェルスクリプト)

```
:  
DSPFILE=mefwrc; export DSPFILE  
PRTFILE=mefprc; export PRTFILE  
:
```

リモート実行

Webサーバマシン上に配置したCOBOLアプリケーションをWebブラウザからリモート実行します。詳細は“[4.6 リモート実行](#)”を参照してください。

第4章 MeFt/Web環境の構築

画面帳票アプリケーションをWeb環境で動作させるにはMeFt/Webが必要です。本章では、MeFt/Web環境の構築について説明します。
ここでの環境構築により、アプリケーションはクライアントで画面の入出力と帳票の印刷を行います



画面定義体をHTMLファイルに変換してWebブラウザに表示する「MeFt/Web HTML変換方式」については、ここでは説明しません。
“NetCOBOL MeFt/Webユーザズガイド (HTML変換方式編)”を参照してください。

4.1 概要

4.1.1 MeFt/Webの概要

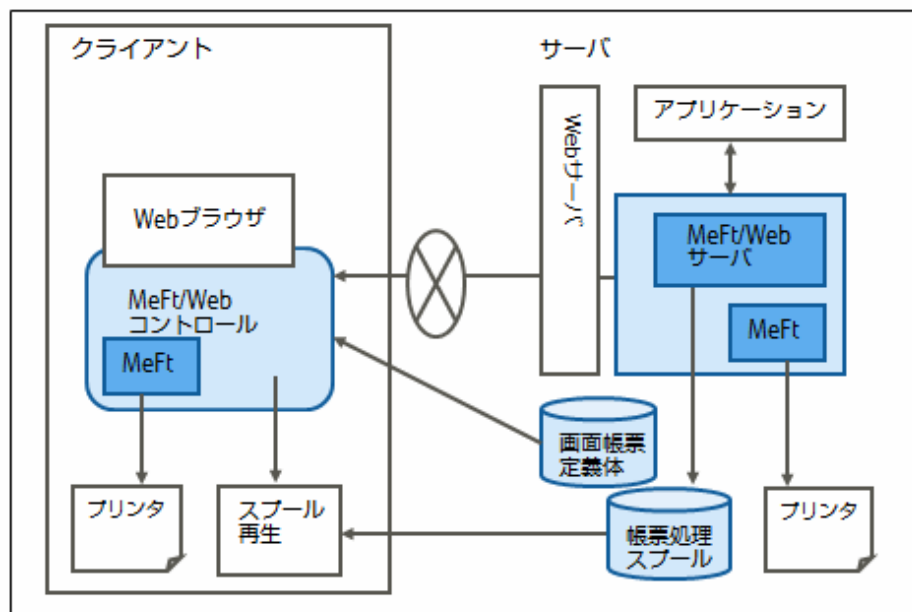
MeFt/Webとは、Webサーバ上で動作するアプリケーションのディスプレイ装置またはプリンタ装置への入出力を、Webブラウザ上で行うことができる通信プログラムです。

MeFt/Webを使用することにより、画面帳票定義体の入出力を行うプログラムをWeb環境で動作させることができますようになります。

MeFt/Webは、サーバ上で動作するWebサーバ連携プログラム(以降、MeFt/Webサーバ)と、クライアント側で動作するActiveXコントロール(以降、MeFt/Webコントロール)から構成されています。

MeFt/Webサーバは、Webサーバを介して、アプリケーションからMeFtに要求された入出力要求を、クライアント側のMeFt/Webコントロールに渡すなどの処理を行っています。

MeFt/Webコントロールは、MeFt/Webサーバからの入出力要求をWebブラウザやプリンタ装置に対して行います。また、MeFt/Webコントロールは、MeFt/Webサーバとの通信処理やMeFt機能がActiveXコントロール化されたものであり、必要時にサーバ上からダウンロードされます。MeFt/Webコントロールからサーバ上のアプリケーションを実行し、画面帳票の入出力をWebブラウザで行うことを「リモート実行」といいます。



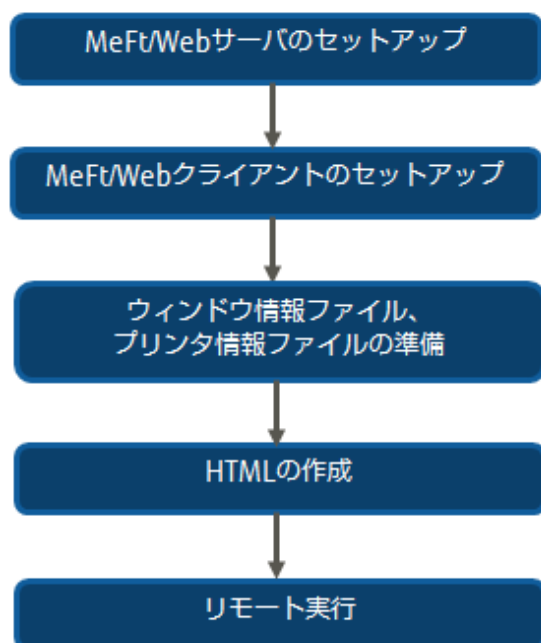
MeFt/Webには、次のような機能があります。

機能名		説明
画面機能	画面処理	リモート実行した利用者プログラムからの画面入出力をWebブラウザ上で行います。

機能名		説明
	ハイパーリンク	項目にURLを設定することができます。
印刷機能	プレビューおよびクライアント印刷機能	印刷イメージをWebブラウザ上に表示したり、クライアントに接続されているプリンタ装置に印刷します。
	サーバ印刷機能	サーバに接続されているプリンタ装置に印刷します。
	スプール機能およびスプール再生機能	利用者プログラムからの印刷要求をサーバ上にスプールしたり、その帳票結果をWebブラウザ上で再生(プレビュー)します。
サービスマネージャ機能		サーバ上の利用者プログラムを起動したり、起動しているプログラムの一覧表示などを行います。

4.1.2 構築作業の流れ

MeFt/Webアプリケーションの構築の流れを次に示します。



4.2 MeFt/Webサーバのセットアップ

以下の環境を例に説明します。

- Webサーバ (Linux(64)) : ホスト名 (IPアドレス) : 192.0.2.0

Interstage Application Server (Interstage HTTP Server)

```

アプリケーション格納ディレクトリ : /example/sample03/
実行可能プログラム : sample3
画面定義体           : DENPY001. smd
帳票定義体           : DENPY002. pmd
オーバーレイ定義体   : DENPY002. ovd
ウィンドウ情報ファイル : mefwrc (*1)
プリンタ情報ファイル  : mefprc (*2)

sample3起動用のシェルスクリプト : sample3. sh
sample3起動用のHTMLファイル     : sample3. html
  
```

*1: MeFt/Webクライアントが使用するため、シフトJISで作成します。

*2: サーバ印刷の場合はサーバのコード系を使用します。クライアント側で印刷する場合はクライアント環境のコード系を使用します。この例では、クライアント側で印刷するため、シフトJISで作成します。

- ・ クライアント(Windows): Internet Explorer(R)

4.2.1 Webサーバの環境設定

4.2.1.1 仮想ディレクトリ(エイリアス)の設定

Webブラウザから、MeFt/WebのCGIおよびHTMLにアクセスできるように、MeFt/Webの運用に必要な仮想ディレクトリをWebサーバに設定します。

Webサーバの設定方法には、環境定義ファイルhttpd.confにMeFt/Web固有の設定を1つずつ直接記述していく方法と、MeFt/Web固有の設定を記述した雛型ファイル(*1)を編集して環境定義ファイルhttpd.confからIncludeする方法の2通りがあります。

ここでは、MeFt/Web固有の設定を記述した雛型ファイルを使用する方法を説明します。

*1: MeFt/Webの動作に必要な仮想ディレクトリの設定内容の雛型は、以下にインストールされます。

```
/opt/FJSVXmeftw/etc/mw_httpd64.conf
```

1. mw_httpd64.confを編集します。

資源が格納されたディレクトリ("/example/sample03")を仮想ディレクトリ("/MeFtWeb64/sample3/")として設定するため、以下の赤字部分を追加します。

```
Alias /MeFtWeb64/sample3/ "/example/sample03/"
-----+
| 追加する
| 仮想ディレクトリの設定
|
|
|-----+

Alias /MeFtWeb64/FGATEWAY "/opt/FJSVXmeftw/MeFtWeb64/FGATEWAY2"
Alias /MeFtWeb64/ "/opt/FJSVXmeftw/MeFtWeb64/"
<Directory "/opt/FJSVXmeftw/MeFtWeb64/">
    Options None
    AllowOverride None
    # Require all granted
</Directory>
ScriptAlias /mw-bin64/f4eswsc0 "/opt/FJSVXmeftw/bin/mw-bin64/f4eswsc0"
<Directory "/opt/FJSVXmeftw/bin/mw-bin64/">
    Options None
    AllowOverride None
    # Require all granted
</Directory>
:
```



注意

apache 2.4をご利用の場合は、“Require all granted”を有効にしてください。

2. Interstage Application Server (Interstage HTTP Server)を利用する場合、環境定義ファイルhttpd.confを編集して、1.で編集したmw_httpd64.confをインクルードします。
Includeディレクティブを使用してhttpd.confに次の1行を追加します。

```
Include /opt/FJSVXmeftw/etc/mw_httpd64.conf
```



参考

httpd.confのインストールパス、および各ディレクティブについては、利用するWebサーバのマニュアルを参照してください。また、Red Hat Enterprise Linuxに付属するApache HTTP Serverを利用する場合の設定方法は、“NetCOBOL ソフトウェア説明書”の“MeFt/Webの環境設定”を参照してください。

4.2.1.2 実行時ユーザの設定

MeFt/Webがリモート実行するプロセスのユーザとCGIの実行ユーザを一致させます。

ここでは、実行ユーザ名を「coboluser」として、以下のファイルのユーザ名を一致させます。

「SysVinit」の場合

- MeFt/Webサーバ起動用のシェル・スクリプト・ファイル(/opt/FJSVXmeftw/etc/rc3.d/S99meftweb64)

```
EXECUSER=coboluser
```

- Webサーバの環境定義ファイル(httpd.conf)

```
User coboluser
```

「Systemd」の場合

/usr/lib/systemd/system/FJSVXmeftwd.serviceファイルを/etc/systemd/systemにコピーします。

- /etc/systemd/system/FJSVXmeftwd.serviceファイル

```
User=coboluser
```

- /opt/FJSVXmeftw/config/FJSVXmeftwd.confファイル

```
MWUSER="coboluser"
```

最後に変更内容を反映するため以下のコマンドを実行します。

```
# systemctl daemon-reload
```

4.2.1.3 Webサーバの再起動時

Webサーバの環境設定を有効にするために、Webサーバを再起動します。Webサーバの再起動については、利用するWebサーバのマニュアルを参照してください。

4.2.2 MeFt/Web動作環境の設定

MeFt/Web の動作環境情報は、mwsetup コマンドで設定します。

mwsetup コマンドを実行する場合は、スーパー・ユーザで行います。

```
# su
# /opt/FJSVXmeftw/bin/mwsetup
```

mwsetup コマンドを実行すると、MeFt/Web 動作環境設定画面が表示されます。変更する動作環境の番号を指定し、画面表示に従って動作環境を設定します。

mwsetup コマンドは、ja_JP.UTF-8 ロケールでのみ使用できます。

MeFt/Web動作環境	
1. 同時実行可能数	9999
2. 利用者プログラムの通信監視時間	0
3. ログ出力有無	0

4. ログ出力ディレクトリ	/var/opt/FJSVXmeftw/log/
5. スプール出力ディレクトリ	/var/opt/FJSVXmeftw/spool/
6. ドキュメント格納ディレクトリ	/opt/FJSVXmeftw/mw-mgr64/document/
7. 利用者プログラムの権限変更	0
q. 動作環境の設定終了	
設定する項目の番号を指定して下さい。	

動作環境設定の各項目の概要を次に示します。詳細は、“MeFt/Webユーザズガイド”の“MeFt/Webの動作環境を設定する”を参照してください。

項目名	説明
同時実行可能数	MeFt/Web サーバからリモート実行する利用者プログラムの同時実行可能数を指定します。無制限を指定する場合には「9999」を指定します。初期値は「9999」です。 ※) Web サーバに指定した「Web サーバが同時に処理することのできる接続数(最大接続数)」を、ここで設定した値よりも大きくする必要があります。Web サーバの最大接続数についての詳細は、使用するWebサーバの説明書を参照してください。
利用者プログラムの通信監視時間	MeFt/Web サーバでは、Web ブラウザからの長時間の無応答またはネットワーク異常などにより、一定の時間(通信監視時間)を超えて利用者プログラムに応答が返らない場合、MeFt の通知コードMEFD_RC_NTIME (N7) を通知して処理を終了させることができます。1分から65535分の間を分単位で指定します。
ログ出力有無	利用者プログラムの標準出力をファイル(meftweb64.stdout) に、利用者プログラムやCOBOL ランタイムシステムの標準エラー出力をファイル(meftweb64.stderr) に格納することができます。 ログ出力を行う場合には「1」を、ログ出力を行わない場合には「0」を指定します。
ログ出力ディレクトリ	「ログ出力有無」を「行う」に指定した場合、ログファイルを格納するディレクトリ名をフルパスで指定します。
スプール出力ディレクトリ	スプール機能を実行した際に印刷データを格納するディレクトリをフルパスで指定します。
ドキュメント格納ディレクトリ	MeFt/Web ドキュメントを格納するディレクトリをフルパスで指定します。
利用者プログラムの権限変更	インストール直後の状態では、MeFt/Web がリモート実行する利用者プログラムはnobodyの権限で起動されます。リモート実行する利用者プログラムの権限をnobody以外に変更する場合には「1」を、利用者プログラムの権限を変更しない場合は「0」を指定します。

この例では、何も変更せずに初期設定のままとします。

4.2.3 利用者プログラムの指定

利用者プログラム指定ファイルに、リモート実行機能で起動する利用者プログラムを指定します。起動する利用者プログラムは、利用者プログラム指定ファイル「/opt/FJSVXmeftw/etc/f4eselst.conf」で指定します。

```
[programs]
/example/sample03/sample3.sh

[resources]
```

利用者プログラム指定ファイルの[programs] セクションに、リモート実行機能で起動する利用者プログラムを指定します。ここでは、起動する利用者プログラムとして、「/example/sample03/sample3.sh」を指定します。

指定方法の詳細は、“MeFt/Webユーザーズガイド”の“利用者プログラム指定ファイルの編集”を参照してください。

4.2.4 MeFt/Web サーバの起動

「SysVinit」の場合

MeFt/Webをインストールすると、MeFt/Webサーバは、システムの起動時に自動的に起動されます。手動でMeFt/Webサーバを起動するには、以下のコマンドを実行します。なお、MeFt/Web サーバの起動は、スーパー・ユーザで行います。

```
# /opt/FJSVXmeftw/etc/rc3.d/S99meftweb64 start
```

なお、起動しているMeFt/Web サーバを停止して、再度起動する場合は以下を実行します。

```
# /opt/FJSVXmeftw/etc/rc3.d/S99meftweb64 stop
# /opt/FJSVXmeftw/etc/rc3.d/S99meftweb64 start
```

「Systemd」の場合

手動でMeFt/Webサーバを起動するには、以下のコマンドを実行します。

```
# systemctl start FJSVXmeftwlogd ( トレースログを採取する場合 )
# systemctl start FJSVXmeftwd
```

MeFt/Webサーバを停止するには、以下のコマンドを実行します。

```
# systemctl stop FJSVXmeftwd
# systemctl stop FJSVXmeftwlogd ( トレースログを採取する場合 )
```

システム起動時にMeFt/Web サーバを自動的に起動するには、以下のコマンドを実行します。

```
# systemctl enable FJSVXmeftwlogd ( トレースログを採取する場合 )
# systemctl enable FJSVXmeftwd
```

システム起動時にMeFt/Web サーバを自動的に起動しないようするには、以下のコマンドを実行します。

```
# systemctl disable FJSVXmeftwlogd
# systemctl disable FJSVXmeftwd
```

4.3 MeFt/Web クライアントのセットアップ

MeFt/Web クライアントには、以下の2 種類があります。

- MeFt/Webプラグイン
- MeFt/Web コントロール

それぞれの違いについては、“MeFt/Webユーザーズガイド”の“MeFt/Web クライアント”を参照してください。

ここでは、MeFt/Web コントロールを使用する方法を説明します。

4.3.1 MeFt/Webコントロールのダウンロード

クライアントマシン上で動作するMeFt/Web コントロールは、ActiveX(R) コントロールです。HTMLのOBJECTタグにMeFt/Web コントロールの格納先を指定すると、自動的にサーバ上からダウンロード／セットアップされます。プログラム起動用のHTMLファイルの詳細は、“[4.5 HTMLの作成](#)”を参照してください。

ここでは、以下のように指定します。

sample3起動用のHTMLファイル(sample3.html)

```
:
<OBJECT
ID="MeFtWeb1"
CLASSID="CLSID:61F12C43-5357-11D0-9EA0-00000E4A0F56"
WIDTH="980" HEIGHT="610"
```

```
CODEBASE="http://192.0.2.0/MeFtWeb64/meftweb.cab#version=12,0,0,2">
</OBJECT>
:
```

MeFt/Web コントロールがバージョンアップ、レベルアップされた場合、CODEBASE に記述されているバージョン情報を更新する必要があります。OBJECTタグの詳細については、“MeFt/Web ユーザーズガイド”の“MeFt/Web コントロールをサーバ上からダウンロードする”を参照してください。

4.4 ウィンドウ情報ファイル、プリンタ情報ファイルの準備

ウィンドウ情報ファイル(mefwrc) およびプリンタ情報ファイル(mefprc)に、ユーザ資源の格納パスを指定します。

この例では、以下のように書き換えます

ウィンドウ情報ファイル(mefwrc)

```
TITLE "SAMPLE3"
*WINSIZECX 80
*WINSIZECY 18
MEDDIR http://192.0.2.0/MeFtWeb64/sample3
MEDSUF smd
```

プリンタ情報ファイル(mefprc)

```
PRTID "SAMPLE3"
PRTFONT FAC
PRTDIALG Y
MEDDIR http://192.0.2.0/MeFtWeb64/sample3
OVLDIR http://192.0.2.0/MeFtWeb64/sample3
MEDSUF pmd
OVLPSUF ovd
```

4.5 HTMLの作成

Webサーバ上のプログラムをリモート実行するには、HTMLを作成する必要があります。

HTMLでは、次に示すような内容を記述します。

- MeFt/Webコントロールの指定
- アプリケーションの起動方法
- Webサーバの指定
- リモート実行するアプリケーションの指定
- リモート実行時の動作に関する指定
- ページ移動時の動作に関する指定

ここでは、サンプルプログラムで提供されているHTMLをコピーし、必要な部分を修正します。

サンプルプログラムのHTMLは次に示す場所にある「sample3.html」です。

```
/opt/FJSVcb164/samples/ja_JP.UTF-8/sample03/sample3.html
```

修正したHTMLを以下に示します。下線付き赤字の部分が修正した箇所です。

```
<HTML>
<HEAD>
<TITLE>sample3</TITLE>
</HEAD>
<BODY>
<INPUT TYPE=BUTTON VALUE="GO!" NAME="GO"><BR>
```

```

<OBJECT
ID="MeFtWeb1"
CLASSID="CLSID:61F12C43-5357-11D0-9EA0-00000E4A0F56"
WIDTH="980" HEIGHT="610"
CODEBASE="http://192.0.2.0/MeFtWeb64/meftweb.cab#version=12,0,0,2">
</OBJECT>
<SCRIPT LANGUAGE="VBScript">
Sub GO_onClick()
MeFtWeb1.hostname = "192.0.2.0"
MeFtWeb1.gatewaypathname = "MeFtWeb64"
MeFtWeb1.pathname = "/example/sample03/sample3.sh"
MeFtWeb1.environment = "MEFTWEBDIR=http://192.0.2.0/MeFtWeb64/sample3"
MeFtWeb1.printmode = 1
MeFtWeb1.submit()
end sub
Sub Window_onUnload()
MeFtWeb1.Quit()
end sub
</SCRIPT>
</BODY>
</HTML>

```

「sample3.html」として、「example/sample03/」に保存します。

このHTMLを元に、記述する内容を説明します。

MeFt/Webコントロールの指定

MeFt/Webの画面を表示するため、MeFt/Webコントロールの指定を行います。

MeFt/Webコントロールの指定は、HTMLのOBJECTタグに記述します。CODEBASEで指定するMeFt/Webコントロールの格納先には、Webサーバのホスト名を記述します。

ここで作成するHTMLでは、以下の記述でMeFt/Webコントロールを指定しています。

```

<OBJECT
ID="MeFtWeb1"
CLASSID="CLSID:61F12C43-5357-11D0-9EA0-00000E4A0F56"
WIDTH="980" HEIGHT="610"
CODEBASE="http://192.0.2.0/MeFtWeb64/meftweb.cab#version=12,0,0,2">
</OBJECT>

```

アプリケーションの起動方法

どのような操作によってアプリケーションを起動するのかを指定します。アプリケーションの起動には、MeFt/Webコントロールのsubmitメソッドを使用します。

本章で作成するHTMLでは、INPUTタグで指定されたボタンが押されたら起動するようにしています。

```

<INPUT TYPE=BUTTON VALUE="GO!" NAME="GO">
:
<SCRIPT LANGUAGE="VBScript">
Sub GO_onClick()
:
MeFtWeb1.submit()
end sub
:
</SCRIPT>

```



参考

HTMLの表示と同時にアプリケーションを起動するには、以下のように記述します。

```
<SCRIPT LANGUAGE="VBScript">
Sub Window_onload()
:
MeFtWeb1.submit()
end sub
:
</SCRIPT>
```

Webサーバの指定

アプリケーションが格納されているWebサーバのホスト名を指定します。

Webサーバのホスト名は、MeFt/Webコントロールのhostnameプロパティで指定します。Webサーバ名を省略することはできません。

ここで作成したHTMLでは、以下の記述でWebサーバを指定しています。

```
MeFtWeb1.hostname = "192.0.2.0"
```

リモート実行するアプリケーションの指定

どのアプリケーションをリモート実行するのかを指定します。

リモート実行するアプリケーション名は、MeFt/Webコントロールのpathnameプロパティにサーバのローカルパスで指定します。アプリケーション名を省略することはできません。

また、アプリケーション実行時の環境変数は、MeFt/Webコントロールのenvironmentプロパティで指定することができます。

ここで作成したHTMLでは、以下の記述でアプリケーション名と環境変数を指定しています。

```
MeFtWeb1.pathname = "/example/sample03/sample3.sh"
MeFtWeb1.environment = "MEFTWEBDIR=http://192.0.2.0/MeFtWeb64/sample3"
```



参考

ここでは、環境変数としてMEFTWEBDIRを指定しています。環境変数MEFTWEBDIRは、クライアント印刷で使用するプリンタ情報ファイルの格納ディレクトリを指定します。詳細は、“MeFt/Webユーザーズガイド”の“クライアント印刷”を参照してください。

また、サーバ印刷用のプリンタ情報ファイルの格納ディレクトリを指定する環境変数MEFTDIRは指定していません。その場合、サーバ印刷時の出力プリンタは、MeFt/Web動作環境の「サーバ印刷用の出力プリンタデバイス名」の設定に従います。

リモート実行時の動作に関する指定

画面の表示形式や印刷先の指定など、リモート実行時の動作に関する指定は、MeFt/Webコントロールの各プロパティで指定します。

ここで作成したHTMLでは、以下の記述でリモート実行時の動作に関する指定を行っています。MeFt/Webコントロールの各プロパティの説明は、“MeFt/Webユーザーズガイド”の“プロパティ”を参照してください。

```
MeFtWeb1.gatewaypathname = "MeFtWeb64"
MeFtWeb1.printmode = 1
```

---“1”: クライアントに接続されているプリンタ装置に印刷

ページ移動時の動作に関する指定

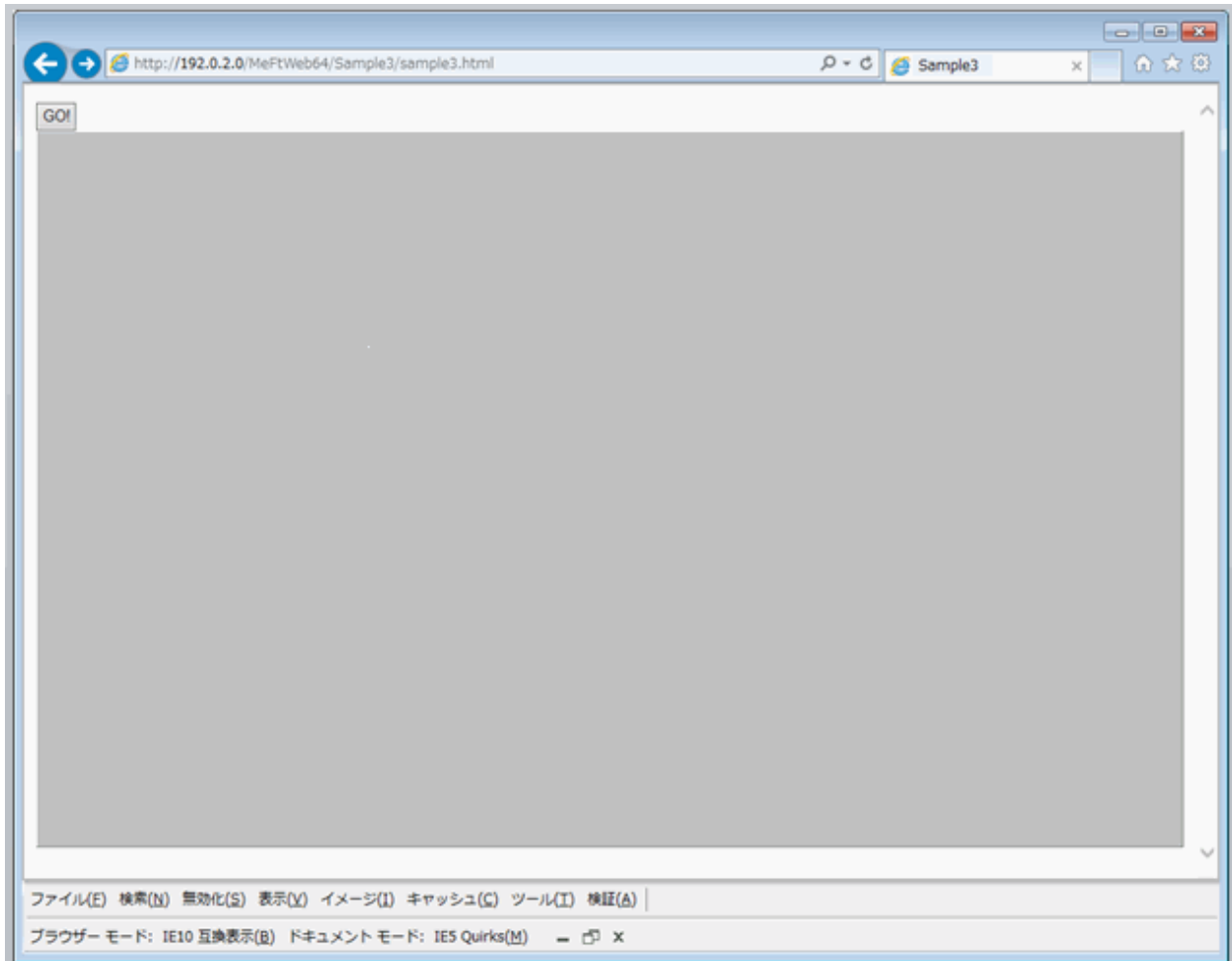
リモート実行中にページが移動された場合の動作は、ページ移動時に発生するWindow_onUnloadイベントの処理として指定します。リモート実行中にページが移動された場合、そのことをアプリケーションに通知するため、MeFt/WebコントロールのQuitメソッドを記述します。

ここで作成したHTMLでは、以下の記述でページ移動時の動作に関する指定を行っています。詳細は、“[4.7 通信が切断されるパターンについて](#)”を参照してください。

```
<SCRIPT LANGUAGE="VBScript">
:
Sub Window_onUnload()
MeFtWeb1.Quit()
```

```
end sub  
:  
</SCRIPT>
```

作成したHTMLをWebブラウザで表示すると、次のようになります。



4.6 リモート実行

Webサーバマシン上に配置したCOBOLアプリケーションをWebブラウザからリモート実行する方法を説明します。



注意

SELinuxを有効にした環境でCOBOLアプリケーションを実行する場合は、SELinuxのセキュリティコンテキストの設定が必要です。詳細は、“MeFt/Webユーザーズガイド”の“SELinux有効環境での注意事項”を参照してください。

4.6.1 HTMLの表示

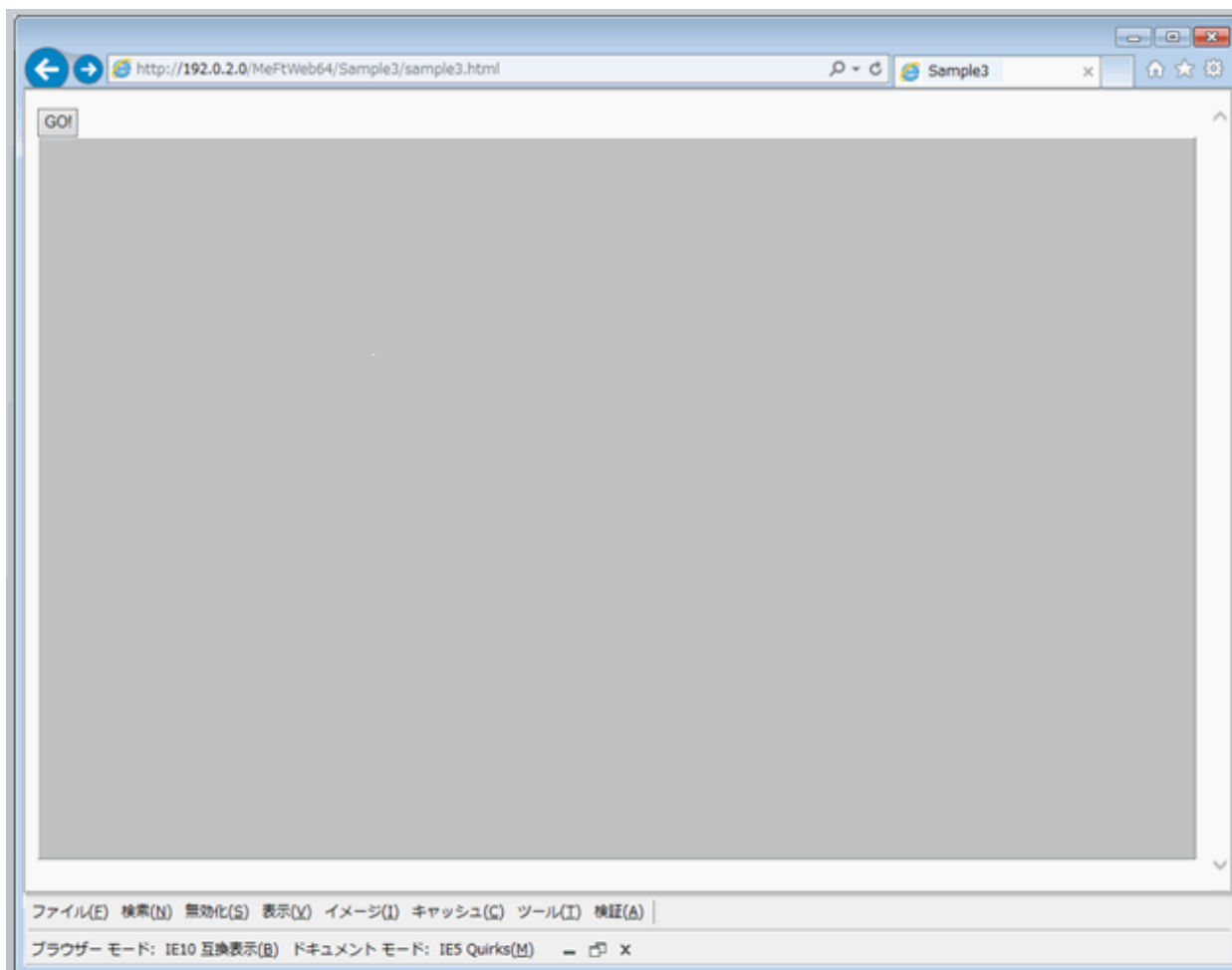
Webブラウザで、MeFt/Webをリモート実行するHTMLを表示するために、次のようなURLを指定します。

```
http://192.0.2.0/MeFtWeb64/sample3/sample3.html
```

リモート実行を行うには、クライアントにMeFt/Webコントロールをダウンロードする必要がありますが、Webブラウザでリモート実行するためのHTMLを表示すると、サーバマシンからMeFt/Webコントロールが自動的にダウンロードされます。

なお、MeFt/Webコントロールがダウンロードされるのは、MeFt/Webサーバへの初回の接続時のみです。2回目以降は、ダウンロードされているMeFt/Webコントロールが使用されます。

MeFt/Webコントロールがダウンロードされると、次のような画面が表示されます。



4.6.2 プログラムの実行

1. HTMLの[GO!]ボタンを押すと、プログラムが起動し、次のような画面がWebブラウザに表示されます。

2014 . 4 . 25 (金)

コード	商品名	数量	単価	金額

PF1 : 計算 PF2 : 次の伝票を入力 合計

PF3 : 終了

ファイル(E) 検索(N) 無効化(S) 表示(V) イメージ(I) キャッシュ(C) ツール(T) 検証(A) |
ブラウザー モード: IE10 互換表示(B) ドキュメントモード: IE5 Quirks(M) 印刷 閉じる

2. 表示された画面に必要な情報を入力します。

2014 . 4 . 25 (金)

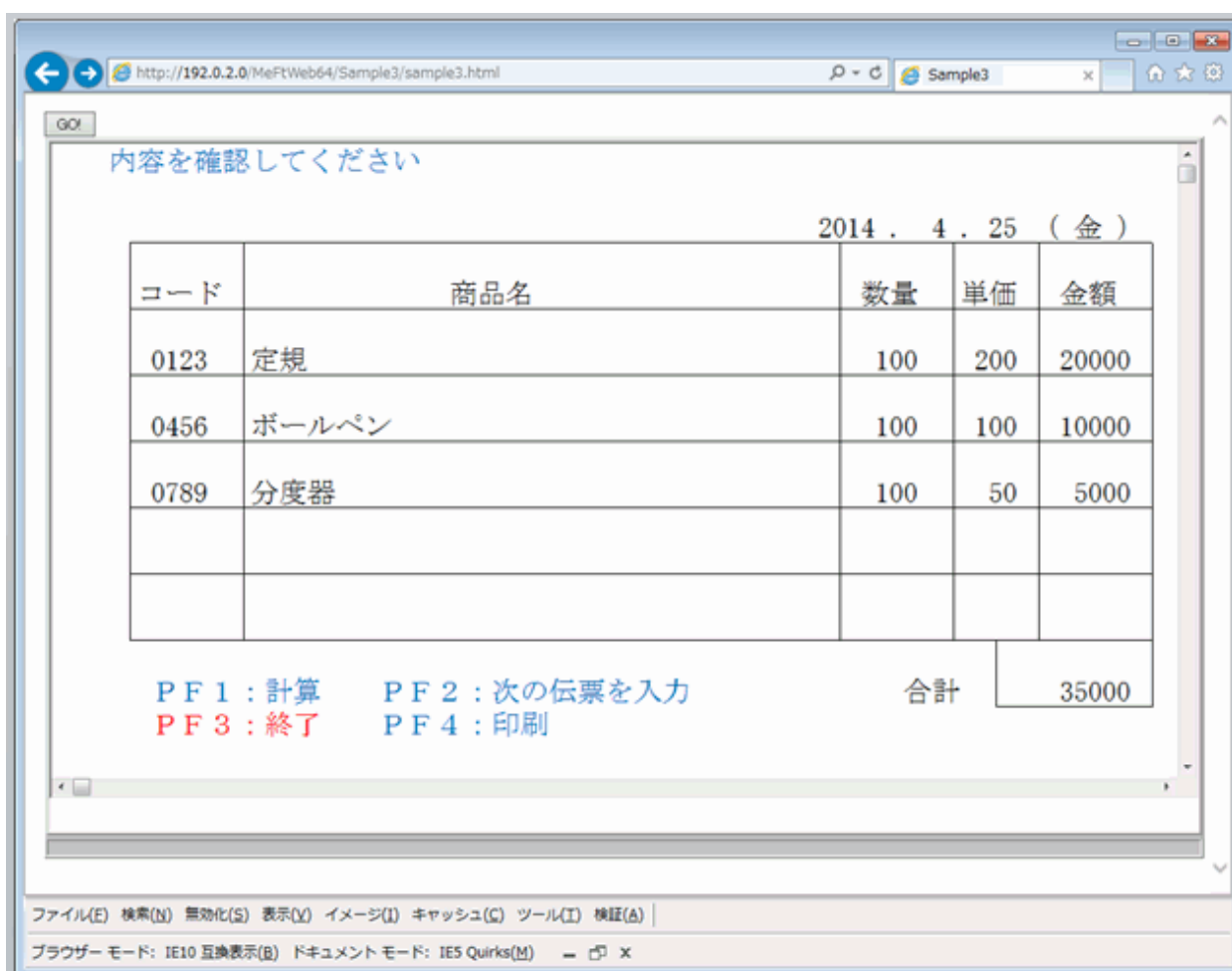
コード	商品名	数量	単価	金額
0123		100		
0456		100		
0789		100		

PF 1 : 計算 PF 2 : 次の伝票を入力 合計

PF 3 : 終了

ファイル(E) 検索(N) 無効化(S) 表示(V) イメージ(I) キャッシュ(C) ツール(T) 検証(A) |
ブラウザー モード: IE10 互換表示(B) ドキュメント モード: IES Quirks(M) 閉じる

3. [F1]キーを押すと、サーバのアプリケーションは計算処理を行います。



4. [F4]キーを押すと、サーバのアプリケーションは印刷処理を行います。

MeFt/Webの印刷機能には次の4つがあります。

- プレビュー
- クライアント印刷
- サーバ印刷
- スプール

それぞれの機能の詳細は、“MeFt/Webユーザズガイド”の“プレビュー機能”、“クライアント印刷”、“サーバ印刷”、“スプール機能”を参照してください。

アプリケーションが印刷処理を行った場合、どのような印刷を行うかはリモート実行するためのHTMLで定義しますが、今回使用するHTMLでは、クライアント印刷が設定されています。

印刷結果

販売注文書				
2014 . 4 . 25 (金)				
コード	商品名	数量	単価	金額
0123	定規	100	200	20000
0456	ボールペン	100	100	10000
0789	分度器	100	50	5000
		0	0	0
		0	0	0
注文合計				35000

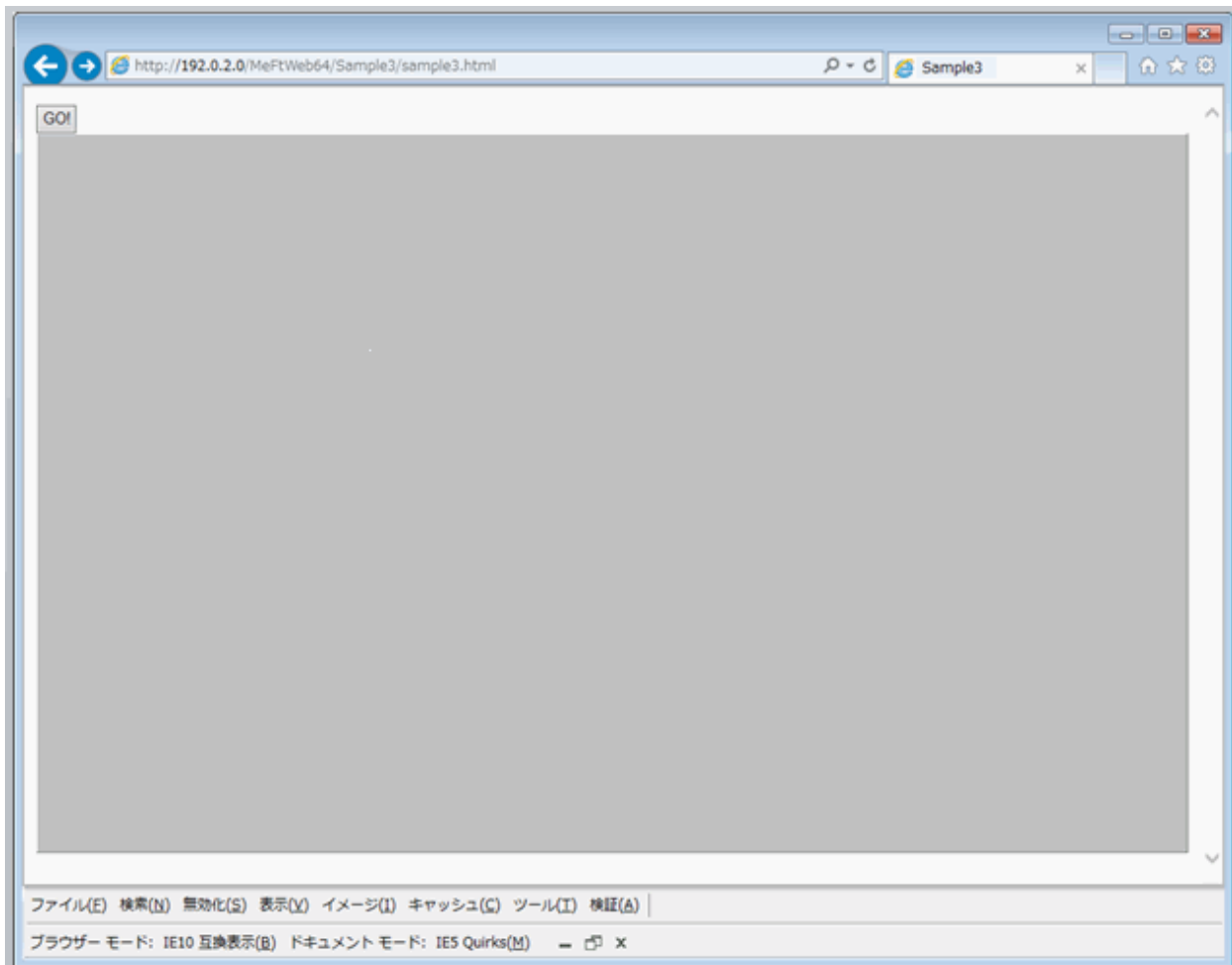
参考

帳票定義体を使用した印刷処理では、プレビューやクライアント印刷などのMeFt/Webの印刷機能を使用できます。

一方、帳票定義体を使用しない印刷処理では、MeFt/Webの印刷機能を使用しないサーバアプリケーションの印刷として処理されます。

プログラムの終了

ここでリモート実行されたアプリケーションでは、入力画面で[F3]キーを押すと、プログラムが終了します。プログラムが終了すると、Webブラウザは、リモート実行前と同じように表示されます。



4.7 通信が切断されるパターンについて

MeFt/Webアプリケーションの運用中に、サーバとクライアントとの間の通信が切断されるパターンとして、次の2つがあります。

- Webブラウザで[戻る]ボタンが押されたり、Webブラウザのアドレスバーで別のURLが指定されたとき
- クライアントマシンの電源が切断されたり、ネットワークが不通になったとき

サーバとクライアントとの間の通信が切断された場合、サーバのアプリケーションはそのことを認識できず、クライアントからの入力を待ちつづける状態となります。しかし、通信切断によりクライアントから応答を返すことができないため、正常にプログラムを終了することができなくなります。プログラムを終了するには強制終了するしかありませんが、ファイルやデータベースが正常にクローズされないため、データに影響が出る可能性があります。

MeFt/Webアプリケーションを構築するにあたっては、このような通信切断時への対応を考慮する必要があります。以下に、とるべき対応を説明します。

Webブラウザで[戻る]ボタンが押されたり、Webブラウザのアドレスバーで別のURLが指定されたとき

Webブラウザで[戻る]ボタンが押されたり、Webブラウザのアドレスバーで別のURLが指定されたりすると、他のページに移動します。その結果、それまで入出力を行っていたページとサーバとの通信が切断されてしまいます。この状況に対応するには、MeFt/WebコントロールのQuitメソッドを使用します。Quitメソッドを使用すると、Webブラウザでイベントが発生したことをアプリケーションに通知することができます。

リモート実行に使用するHTMLで、Webブラウザのページが遷移したときに発生するWindow_onUnloadイベントの処理としてMeFt/WebコントロールのQuitメソッドを実行するように記述すると、Webブラウザで[戻る]ボタンが押されるなどしてページが移動した場合、Quitメソッドが実行されてアプリケーションにコードが通知されます。

HTMLでのQuitメソッドの記述例を以下に示します。

```

<HTML>
:
<OBJECT
ID="MeFtWeb1"
:
</OBJECT>
<SCRIPT type="text/javascript">
function Window_onUnload() {
MeFtWeb1.Quit();
} :
</SCRIPT>
</HTML>

```

COBOLアプリケーションには、表示ファイルのFILE STATUS句に指定された4桁のデータ名の領域に「90N8」で通知されます。それを判定することによってページが移動されたことを知ることができるので、ファイルのクローズやデータベースの切断などの後処理を行うようにします。

MeFt/WebコントロールのQuitメソッドは、“MeFt/Webユーザーズガイド”の「利用者プログラムの中断(Quit)」を参照してください。

クライアントマシンの電源が切断されたり、ネットワークが不通になったとき

クライアントマシンの電源が切断されたり、ネットワークが不通になったりして、サーバとクライアントとの間の通信が切断された状況に対応するには、MeFt/Webサーバの通信監視時間の機能を使用します。

MeFt/Webサーバに通信監視時間を設定すると、設定された通信時間を越えてクライアントからレスポンスがなかった場合、MeFt/Webサーバからアプリケーションにコードを通知します。COBOLアプリケーションには、表示ファイルのFILE STATUS句に指定された4桁のデータ名の領域に「90N7」で通知されます。それを判定することによって通信監視時間を超えて応答がなかったことを知ることができるので、ファイルのクローズやデータベースの切断などの後処理を行うようにします。

MeFt/Webの通信監視時間の設定は、“4.2.2 MeFt/Web動作環境の設定”、および“MeFt/Webユーザーズガイド”の「MeFt/Webの動作環境を設定する」を参照してください。



参考

設定される通信監視時間が長すぎると、サーバとクライアントとの間の通信が切断されているにもかかわらずアプリケーションは起動し続けることになり、サーバの負荷が高まります。

一方、設定される通信監視時間が短すぎると、画面での作業に時間がかかった場合、突然アプリケーションが終了することになってしまいます。

通信監視時間は、業務の内容に応じて適切な値を設定してください。

第5章 効率のよいプログラムのテクニック

ここでは、プログラムの実行時間を短縮するための細かい注意点を述べます。

プログラムは、効率が良いことの他に、読みやすく、拡張しやすいことも必要です。しかし、これから述べる項目の中には、読みやすさに逆行するものもあります。

プログラム作成時には、以下の知識を念頭において、プログラムを読みやすく作り、実行時命令統計機能を使ってボトルネックを発見し、ボトルネックになっている一連の文に対して再度効率向上のための修正を加える、という方法をおすすめします。

また、細かいコーディング技術を駆使するより、プログラムのアルゴリズムを検討する方が、効率を向上させる程度が大きいことがしばしばあります。細部の検討に入る前に、まず、アルゴリズムを改善できないかを考えることも重要です。

5.1 一般的なテクニック

5.1.1 作業場所節の項目

- レコードを設計する際は、よく使われる項目や関連のある項目を一か所に集めて定義するようにします。よく使われる項目が数キロバイト以上の大きな項目には含まれるような設計は避けてください。
- 転記しても参照されないまま再転記されるような、不要な転記は避けてください。
例えば、集団項目全体に空白文字を転記した後で、改めて個々の項目に別の値を転記するのではなく、必要な項目だけに空白詰めを行ってください。
- 作業場所節にある項目で、プログラム実行中に値を変更する必要のないものは、VALUE句で初期値を設定してください。

5.1.2 ループの最適化

ループの中では、特に、COBOLの文では見かけ上わからない添字計算や、データの属性の変換など、目的コードの生成を極力抑えるような配慮が必要です。

- ループの中で実行しなくてもいい文はループの外に出してください。
- ループの中では、データ名による添字付けを避け、指標名による添字付けを用いてください。
- ループの中で数字転記や数字比較などに用いる項目は、ループの外であらかじめ最適な属性の項目に移してください。

5.1.3 複合条件の判定順序

ANDだけ、またはORだけで結ばれた複合条件は、括弧がない限り左から右へ順に評価されます。以下の様を書く、平均実行時間を短縮することができます。

- ORで結ばれている場合は、真になりやすい条件を先に書く
- ANDで結ばれている場合は、偽になりやすい条件を先に書く

5.2 データ項目の属性を理解して使う

5.2.1 英数字項目と数字項目

英数字項目が使用できる場所は、数字項目でなく、英数字項目を使ってください。

数字項目は、その中に入っている数値が意味を持っています。

例えば、PIC S9 DISPLAYの項目のビットパターンがX'39'でもX'49'でも、数値としては等しく、共に+9を示すものとみなされます。このため、比較などの目的コードは、英数字項目に比べて遅くなります。

5.2.2 USAGE DISPLAYの数字項目(外部10進項目)

- 各文字位置には、文字"0"～"9"(16進表記でX"30"～X"39")が入ります。最後の文字の先頭4ビットは符号を表し、数値が正の場合はX"4"、負の場合はX"5"が入ります。

- ・ 印字表示用として使用してください。演算や比較に使用したときの処理速度は、数字項目の中で最も遅く、使用領域も最も大きくなります。

5.2.3 USAGE PACKED-DECIMALの数字項目(内部10進項目)

- ・ 4ビットで1桁の数値を表し、最後の4ビットは符号を表します。数値が正の場合はX"C"、負の場合はX"D"、符号なしの場合はX"F"が入ります。
- ・ 演算や比較に使用したときの処理速度は、外部10進項目よりは速く、2進項目よりは遅くなります。

5.2.4 USAGE BINARY/COMP/COMP-5の数字項目(2進項目)

- ・ COMP-5の内部表現形式はシステムのエンディアンに従います。BINARYとCOMPは同義で、システムのエンディアンに従わず、常にビッグエンディアンの内部表現形式になります。
- ・ 表示を目的としない演算、添字に適しています。演算や比較は外部10進項目および内部10進項目より速く、リトルエンディアン・システムではBINARYよりCOMP-5が速くなります。

5.2.5 数字項目の符号

数字項目には、その項目に値を転記する時に絶対値をとる必要がある場合を除き、PICTURE句でSを指定してください。符号をつける場合、SIGN LEADINGやSIGN SEPARATEは指定しないでください。

- ・ Sの指定がないと、転記する時に絶対値をとるための目的コードが生成されます。
- ・ SIGN LEADINGやSIGN SEPARATEの指定をすると、符号処理のための余分な命令が生成されます。

5.3 数字転記・数字比較・算術演算の処理時間を短くする

5.3.1 属性

- ・ 数字転記、数字比較、算術演算では、作用対象のUSAGE句を統一してください。
- ・ 数字転記、数字比較、加減算では、作用対象の小数部桁数を一致させてください。乗除算では、中間結果の小数部桁数と受取り側項目の小数部桁数を一致させてください。
 - ― 一致していないと、演算や比較のたびに、一致させるための変換や桁合せのための目的コードが生成されます。
 - ― 乗算 $C=B*A$ では $dc=db+da$ を、除算 $C=B/A$ では $dc=db-da$ を満たすようにすると、桁合せは発生しません。(da,db,dcはそれぞれA,B,Cの小数部桁数を表します)
- ・ 算術式では、属性が同じもの同士の演算が多くなるような順に、演算を行ってください。

5.3.2 桁数

桁数は、必要以上に大きくとらないでください。

一般的に、演算や比較の時間は、桁数が多いほど長くなります。

5.3.3 べき乗の指数

べき乗の指数は、整数の定数が最も適しています。次に適しているのは、整数項目です。

整数でない指数が指定されると、COBOLランタイムシステムによる浮動小数点演算となるので、効率は極めて悪くなります。

5.3.4 ROUNDED指定

ROUNDED指定の使用は、必要最小限にしてください。

ROUNDED指定をすると、演算結果が1桁余分に求められ、正負の判定と四捨五入を行う目的コードが生成されます。

5.3.5 ON SIZE ERROR指定

ON SIZE ERROR指定の使用は、必要最小限にしてください。

ON SIZE ERROR指定すると、桁あふれを判定するために以下のような目的コードが生成されます。

- ・ 演算結果が2進で求まる場合
絶対値をとるなどして受取り側項目に収まる最大値との比較
- ・ 演算結果が内部10進で求まる場合
受取り側項目の文字位置を超える部分とゼロとの比較

5.3.6 TRUNCオプション

- ・ TRUNCオプションの使用が必要最小限になるように、プログラムを設計してください。
- ・ TRUNCオプションを指定した場合、2進項目間の転記における切り捨ては、 $10^{**}n$ (n は受取り側項目の桁数)で除算し、剰余を求めて行っています。したがって、2進項目を多用するプログラムでは、TRUNCオプションを指定すると、大幅に効率が悪くなります。
- ・ NOTRUNCオプションを指定する場合、受取り側項目に文字位置を超える値が入らないようにプログラムを設計しなければなりません。入力データによってそのような問題が起こる可能性がある場合は、不当な入力データを除外するプログラムに変更した上で、NOTRUNCオプションを指定してください。

5.4 英数字転記・英数字比較を効率よく行う

5.4.1 境界合せ

機種によって異なりますが、英数字項目も左端を4バイトまたは8バイト境界に合わせると、一般に効率がよくなります。ただし、英数字項目に対して指定されたSYNCHRONIZED句は注釈とみなされるので、使用しない項目を定義して境界を合わせるようにしてください。境界合せによって全体の使用領域は大きくなります。境界合せは、よく使われる項目を対象にしてください。

5.4.2 項目長

- ・ 英数字転記では、送出し側項目長が受取り側項目長より大きい・等しい時、効率よく実行できます。英数字比較では、両方の項目長が等しい時、効率よく実行できます。
一方が定数の時は、その長さを他方の項目長に合わせると、効率よく実行できます。
- ・ 上記は、大きな項目(数百バイト以上)の場合、あてはまりません。

5.4.3 転記の統合

複数個のMOVE文が、それらの項目を含む集団項目のMOVE文にまとめられるときは、1個の集団項目のMOVE文にしてください。

5.5 入出力におけるテクニック

5.5.1 SAME RECORD AREA句

SAME RECORD AREA句は、2つ以上のファイルでレコード領域の内容を共有したい場合や、WRITE文の実行後もレコードを使用する必要がある場合に限って指定してください。

SAME RECORD AREA句が指定されたファイルのREAD文およびWRITE文は、レコード領域とバッファ領域の間でレコードの転送を行うため、効率が悪くなります。

5.5.2 ACCEPT文、DISPLAY文

ACCEPT文(DATE、DAY、TIME指定を除く)およびDISPLAY文は、少量のデータの入出力のみに用いてください。

これらの文は、READ文およびWRITE文よりも一般的に効率が悪くなります。

5.5.3 OPEN文、CLOSE文

OPEN文およびCLOSE文は、非常に複雑な内部処理を伴う文であるため、1つのファイルに対するOPEN文およびCLOSE文の実行回数は、必要最小限に抑えてください。

5.6 プログラム間連絡におけるテクニック

5.6.1 副プログラムの分割の基準

1つのシステムを多数のプログラムから構成する場合は、必要以上に小さい副プログラムに分割しないことをおすすめします。

- 副プログラムの呼出しから復帰までに、静的構造の場合でも最低数10ステップの機械文が実行されます。したがって、小さい副プログラムでは、このステップ数が相対的に大きな比重を占めることになり、効率を悪化させてしまいます。副プログラムの手続き部が数百行以上あれば、効率は悪化しません。
- 目的プログラムは初期化・終了ルーチンや作業領域などを必ず持っているので、小さい副プログラムに分割すると全体の領域が多く必要になります。

5.6.2 動的プログラム構造と動的リンク構造

動的プログラム構造(CALL一意名、またはDLOADオプションを指定して翻訳したCALL定数を用いるプログラム構造)は、非常に大きなシステムで、仮想記憶を節約するために仮想記憶上から副プログラムを削除する場合以外には使用しないでください。

動的リンク構造で済むときは、動的リンク構造を使うことをおすすめします。

- 動的プログラム構造では、副プログラムがローディングされた後も、副プログラムが既にローディングされているかどうかを調べるサーチ処理や、プログラム名のチェックが、呼出しのたびに行われます。そのため、オーバーヘッドは、静的プログラム構造より大きくなってしまいます。
- 動的リンク構造では、副プログラムがローディングされた後は、呼出しは直接行われます。そのため、オーバーヘッドは、静的リンク構造の場合よりわずかに多い程度です。

5.6.3 CANCEL文

動的プログラム構造を使う場合、CANCEL文は必要最小限に抑えてください。

5.6.4 パラメタの個数

CALL文のUSING指定、およびENTRY文またはPROCEDURE DIVISIONのUSING指定にパラメタを記述すると、個々のパラメタについてアドレスの設定が行われます。したがって、パラメタは可能な限り集団項目にまとめ、USING指定での個数を少なくする方が、効率がよくなります。

5.7 デバッグ機能を使用する

プログラムの誤りを発見する手段として、TRACE機能、CHECK機能、COUNT機能を提供しています。これらの機能をプログラムの運用前に使用することでトラブルの発生防止につながります。なお、運用時には以下の点にご注意ください。

- TRACE機能では、トレース情報の採取など、COBOLプログラムで記述した以外の処理を行います。そのため、TRACE機能使用時には、プログラムのサイズが大きくなり、実行速度も遅くなります。TRACE機能は、デバッグ時にだけ使用し、デバッグ終了後には、翻訳オプションNOTRACEを指定して再翻訳してください。
- COUNT機能では、COUNT情報の採取など、COBOLプログラムで記述した以外の処理を行います。そのため、COUNT機能使用時には、プログラムのサイズが大きくなり、実行速度も遅くなります。COUNT機能は、デバッグ時にだけ使用し、デバッグ終了後には、翻訳オプションNOCOUNTを指定して再翻訳してください。
- CHECKオプションの指定によって、実行時間が2倍以上遅くなることがあります。運用時にCHECK(NUMERIC)を有効にする場合は、可能な限り10進項目を2進項目に変更すると、CHECKオプションによる性能劣化を防ぐことができます。
- CHECK機能およびTRACE機能は、実行時オプションを指定することで、機能を抑制することができます。デバッグ時、一時的にこれらの機能を無効にしたい場合に有効です。

5.8 数字項目の標準規則

ここでは、COBOLプログラムで数字データを扱う上での標準的な規則を述べます。

5.8.1 10進項目

10進項目の入力

- 入力レコード中に10進項目が含まれている場合、誤った内容表現のデータが入らないように注意してください。正しいビットパターンが入っているかどうかを確かめるには、字類条件(IF ... IS NUMERIC)を使います。コンパイラは、READ文の実行時に、10進項目のビットパターンを検査しません。
- 10進項目を含む集団項目へCORRESPONDING指定のない転記を行う場合、誤ったビットパターンが入らないよう注意してください。この場合も、コンパイラは検査を行いません。

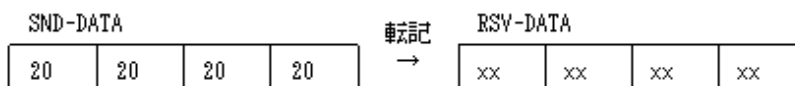
10進項目に誤ったビットパターンが入った場合

- 外部10進項目のゾーン部(符号を持つバイトを除く)が16進数の3でない場合、誤りです。
- 10進項目の数字部が許されるビットパターン(16進数の0~9)でない場合、この項目を転記、演算または比較などに使用すると、結果は規定されません。

誤ったプログラム例

英数字項目から数字項目の転記は数字転記になり、英数字項目を符号なし10進項目にみなして転記します。よって、(a)のSND-DATAはPIC 9(4)とみなし翻訳されます。空白が入っている場合などの不正な値は誤りとなり、結果は予測できません。(a)のMOVE文が予測できないため、(b)の比較結果も予測できません。

```
01 SND-DATA PIC X(4).  
01 RSV-DATA PIC 9(4).  
...  
MOVE SPACE TO SND-DATA  
...  
MOVE SND-DATA TO RSV-DATA ... (a)  
IF RSV-DATA = SPACE THEN ... (b)
```



正しいプログラム例

10進項目に不正な値が設定される可能性がある場合は、(c)のように字類条件(IS NUMERIC)を使用し、正しい値が格納されていることを確認してから使用します。

```
01 SND-DATA PIC X(4).  
01 RSV-DATA PIC 9(4).  
...  
MOVE SPACE TO SND-DATA  
MOVE 0 TO RSV-DATA  
...  
IF SND-DATA IS NUMERIC THEN ... (c)  
MOVE SND-DATA TO RSV-DATA  
ELSE  
DISPLAY NC"データ異常" SND-DATA  
END-IF
```

5.8.2 2進項目

2進項目の値の範囲

2進項目は、一般に、PICTURE句で示された値の範囲より大きい絶対値をもつ値を含むことができます。
NOTRUNC指定の場合、2進項目への転記の際にPICTURE句に合わせた切り捨てが行われなかった結果、正の値が負の値になることもあります。

[参照]“NetCOBOLユーザーズガイド”の“TRUNCオプション”

2進項目のけた数の扱い

2進項目は、PICTURE句より大きい値を含むことができますが、これをDISPLAY文で参照した時は、PICTURE句で示された桁数だけ表示されます。

ON SIZE ERROR指定、またはNOT ON SIZE ERROR指定が記述された演算文では、PICTURE句の指定を超えた値を格納しようとしているかどうかの判定が行われます。

一般に、コンパイラは、2進項目の値はPICTURE句で示された範囲内にあることを前提としてコンパイルを行うため、PICTURE句より大きい値を持つ2進項目を演算などに使用すると、場合によっては異常終了を起こすことがあります。

5.8.3 浮動小数点項目

固定小数点への変換

演算の結果が浮動小数点で求まり、これを固定小数点項目に格納する場合、変換の誤差が最小になるように格納されます。同様に、浮動小数点項目から固定小数点項目へ転記する場合も、変換の誤差が最小になるような値が格納されます。

5.8.4 乗除算の混合時の小数部桁数

次のプログラムの[1]と[2]を比較します。

```
77 X PIC S99 VALUE 10.  
77 Y PIC S9 VALUE 3.  
77 Z PIC S999V99.  
    COMPUTE Z = X / Y * 100.    [1]  
    COMPUTE Z = X * 100 / Y.    [2]
```

[1]の答はZ=333.00、[2]の答はZ=333.33となります。

この違いは、除算を行うタイミングによって発生します。

どちらも、除算の結果はXとZの小数部桁数の大きい方、すなわち小数第2位まで求められます。[1]では、X/Yが3.33と求められ、これが100倍されます。[2]ではX*100の中間結果である1000がYで割られ、333.33が求まります。

よって精度のよい結果を求めるには、[2]のように、乗算を先に除算を後に行ってください。

5.8.5 絶対値がとられる転記

- 受取り側項目が符号なし数字項目である転記の場合、送出し側項目の絶対値が、受取り側項目に格納されます。
- 符号付き数字項目から英数字項目への転記の場合、送出し側項目の絶対値が、受取り側項目に格納されます。

5.9 注意事項

外部ブール項目のビットパターン

- 外部ブール項目の内容は、16進数で30または31です。それ以外は許されません。
- 0～6ビットに不適当な値を持つ外部ブール項目を比較や演算に使用したときの結果は、規定されません。
- 不適当な値を持つ可能性がある場合は、外部ブール項目を字類条件により検査してください。

レコード領域の参照

- レコード領域は、OPEN文の実行後、またはCLOSE文実行前のみ参照することができます。
- 整列併合用ファイルのレコードは、入力手続きまたは出力手続き内でのみ参照することができます。

第6章 サンプルプログラム

ここでは、COBOLの各機能を使用したsampleプログラムについて説明します。

サンプルプログラムは、次の場所の、各ロケール名のディレクトリに格納されます。コピーしてご利用ください。

```
/opt/FJSVcbl64/samples
```

- [6.1 標準入出力を使ったデータ処理\(sample01\)](#)
- [6.2 行順ファイルと索引ファイルの操作\(sample02\)](#)
- [6.3 表示ファイル機能を使ったプログラム\(sample03\)](#)
- [6.4 COBOLプログラム間の呼出し\(sample04\)](#)
- [6.5 コマンド行引数の受取り方\(sample05\)](#)
- [6.6 環境変数の操作\(sample06\)](#)
- [6.7 印刷ファイルを使ったプログラム\(sample07\)](#)
- [6.8 印刷ファイルを使ったプログラム\(応用編\)\(sample08\)](#)
- [6.9 FORMAT句付き印刷ファイルを使ったプログラム\(sample09\)](#)
- [6.10 オブジェクト指向プログラム\(初級編\)\(sample10\)](#)
- [6.11 データベース機能を使ったプログラム\(sample16\)](#)
- [6.12 データベース機能を使ったプログラム\(応用編\)\(sample17\)](#)
- [6.13 COBOLファイルアクセスルーチンのサンプル](#)

6.1 標準入出力を使ったデータ処理(sample01)

ここでは、この製品で提供されているサンプルプログラム-sample01-について説明します。

sample01では、COBOLの小入出力を使って、標準入力からデータを入力したり、標準出力にデータを出力したりするプログラムの例を示します。

小入出力の使い方の詳細は、“NetCOBOL ユーザーズガイド”の“小入出力”を参照してください。

概要

標準入力からアルファベット1文字(小文字)を入力し、入力したアルファベットで始まる単語を標準出力に出力します。

提供プログラム

- sample1.cob (COBOLソースプログラム)
- Makefile (メイクファイル)

使用しているCOBOLの機能

- 小入出力機能

使用しているCOBOLの文

- ACCEPT文
- DISPLAY文
- EXIT文
- IF文
- PERFORM文

翻訳、リンク、実行方法

- cobolコマンドを利用する場合

```
$ cobol -M -o sample1 sample1.cob
最大重大度コードは I で、翻訳したプログラム数は 1 本です。
$ ./sample1
アルファベットを 1 文字（小文字）入力してください。=>    a
apple
$
```

- Makefileを利用する場合

```
$ make

$ ./sample1
アルファベットを 1 文字（小文字）入力してください。=>    a
apple
$
```

6.2 行順ファイルと索引ファイルの操作(sample02)

ここでは、この製品で提供されているサンプルプログラム-sample02-について説明します。

sample02では、エディタを使って作成したデータファイル(行順ファイル)を読み込み、マスタファイル(索引ファイル)を作成するプログラムの例を示します。

行順ファイルおよび索引ファイルの使い方の詳細は、“NetCOBOL ユーザーズガイド”の“ファイル処理”を参照してください。

概要

商品コード、商品名、および単価が入力されているデータファイル(行順ファイル)を読み込み、商品コードを主レコードキー、商品名を副レコードキーとする索引ファイルを作成します。

提供プログラム

- sample2.cob(COBOLソースプログラム)
- datafile(データファイル)
- Makefile(メイクファイル)

使用しているCOBOLの機能

- 行順ファイル(参照)
- 索引ファイル(創成)

使用しているCOBOLの文

- CLOSE文
- EXIT文
- GO TO文
- MOVE文
- OPEN文
- READ文
- WRITE文

翻訳、リンク、実行方法

- cobolコマンドを利用する場合

```
$ cobol -M -o sample2 sample2.cob
最大重大度コードは1で、翻訳したプログラム数は1本です。
$ INFILE=datafile; export INFILE
$ OUTFILE=master; export OUTFILE
$ ./sample2
$
```

- Makefileを利用する場合

```
$ make

$ INFILE=datafile; export INFILE
$ OUTFILE=master; export OUTFILE
$ ./sample2
$
```

プログラムの実行終了後、商品コードをキーとする索引ファイル(master)がsample02のディレクトリに作成されます。

索引ファイル(master)が正しく作成されたことを確認する場合は、ファイルユーティリティを使用してください。ファイルユーティリティの表示コマンドでは、索引ファイルのレコードを表示することができます。使用方法の詳細は、“NetCOBOL ユーザーズガイド”の“ファイルユーティリティ”を参照してください。



参考

ソースプログラムのASSIGN句にファイル識別名を記述した場合、プログラムを実行する前にファイルを割り当てる必要があります。ファイルを割り当てるには、ファイル識別名を環境変数名として、使用するファイルの名前をその環境変数に設定します。このsampleでは、データファイル(行順ファイル)を割り当てる環境変数はINFILE、マスタファイル(索引ファイル)を割り当てる環境変数はOUTFILEです。



注意

すでにmasterというファイルが存在する場合、そのファイルの内容は失われます。

6.3 表示ファイル機能を使ったプログラム(sample03)

ここでは、この製品で提供するサンプルプログラム-sample03について説明します。

sample03では、表示ファイル機能を使って、画面から入力したデータを画面に出力し、さらにデータを印刷装置に出力するプログラムの例を示します。画面入出力を行う場合の、表示ファイル機能の使い方は“NetCOBOL ユーザーズガイド”の“表示ファイル機能(画面入出力)の使い方”を、印刷を行う場合の表示ファイル機能の使い方は、“NetCOBOL ユーザーズガイド”の“表示ファイル(帳票印刷)の使い方”を参照してください。なお、このプログラムを実行するには、MeFtおよびMeFt/Webが必要です。

概要

ディスプレイ装置に表示された入力画面に商品コードおよび個数を入力すると、商品コードをキーにマスタファイルを検索し、商品名、単価および金額を求め、さらに合計金額を計算し、ディスプレイ装置に表示します。また、帳票を印刷装置に出力します。

提供プログラム

- sample3.cob(COBOLソースプログラム)
- denpyou.cob(COBOLソースプログラム)
- URIAGE.cbl(登録集原文)
- SHOHINM.cbl(登録集原文)
- DENPYO01.smd(画面定義体)

- DENPYO02.pmd(帳票定義体)
- DENPYO02.ovd(オーバーレイ定義体)
- Mefwrc(ウィンドウ情報ファイル)
- Mefprc(プリンタ情報ファイル)
- sample3.html(起動用のHTMLファイル)
- sample3.sh(起動用のシェルスクリプト)
- Makefile(メイクファイル)

使用しているCOBOLの機能

- 表示ファイル機能(画面入出力)
- 表示ファイル機能(帳票印刷)
- 索引ファイル(参照/創成)
- 登録集の取込み
- 小入出力機能

使用しているCOBOLの文

- OPEN文
- READ文
- WRITE文
- CLOSE文
- PERFORM文
- DISPLAY文
- IF文
- MOVE文
- GO TO文
- EXIT文
- COPY文

プログラムを実行する前に

- MeFt/Webのセットアップを行い、使用できる状態にしておいてください。[参照]“[4.2 MeFt/Webサーバのセットアップ](#)”、“[4.3 MeFt/Webクライアントのセットアップ](#)”
- sample02で作成されるマスタファイルを使用します。
“[6.2 行順ファイルと索引ファイルの操作\(sample02\)](#)”を実行しておきます。
- 起動用のHTMLファイルおよび起動用のシェルスクリプトの環境設定を、使用する環境にあわせて修正してください。

起動用のHTMLファイル(sample3.html)

- CODEBASE
- MeFtWeb1.hostname
- MeFtWeb1.pathname
- MeFtWeb1.environment

起動用のシェルスクリプト(sample3.sh)

- PATH
- DATAFILE
- SALEFILE
- MEFTDIR
- 実行可能プログラムのパス

翻訳、リンク

- cobolコマンドを利用する場合

```
$ cobol -M -o sample3 sample3.cob
最大重大度コードは I で、翻訳したプログラム数は 1 本です.
$
```

- Makefileを利用する場合

```
$ make
$
```

実行

WWWサーバマシン上に配置したCOBOLアプリケーションをWWWブラウザからリモート実行します。詳細は、“[4.6 リモート実行](#)”を参照してください。



- SELinuxを有効にした環境でサンプルプログラムを実行する場合は、SELinuxのセキュリティコンテキストの設定が必要です。詳細は、“MeFt/Webユーザーズガイド”の“SELinux有効環境での注意事項”を参照してください。
- 画面帳票定義体(DENPYO01.smd、DENPYO02.pmd)をテキストエディタなどで開いて保存すると、ファイルの内容が破壊されることがあります。画面帳票定義体を更新する場合には、FORMまたはPowerFORMのエディタ以外は使用しないでください。

6.4 COBOLプログラム間の呼出し(sample04)

ここでは、この製品で提供されているサンプルプログラム-sample04-について説明します。

sample04では、COBOLプログラムから、サブルーチンを呼び出すプログラムの例を示します。

COBOLプログラムの呼出し関係については、“NetCOBOL ユーザーズガイド”の“COBOLプログラムからCOBOLプログラムを呼び出す”を参照してください。

概要

商品コード、商品名、および単価が格納されているマスタファイルの内容を印刷可能文字に変換して、作業用のテキストファイル“work”（カレントディレクトリに作成）に格納します。次に、印刷処理サブプログラム(print.cob)を呼出し、作業用テキストファイルの内容を印刷します。

提供プログラム

- sample4.cob(COBOLソースプログラム)
- print.cob(COBOLソースプログラム)
- s_rec.cbl(登録集原文)
- master(マスタファイル)
- Makefile(メイクファイル)
- COBOL.CBR(実行用初期化ファイル)

使用しているCOBOLの機能

- ・ プログラム間連絡機能
- ・ 登録集の取込み
- ・ 小入出力機能
- ・ 印刷ファイル
- ・ 索引ファイル(参照)
- ・ 行順ファイル(創成)
- ・ 実行時パラメタの受渡し

使用しているCOBOLの文

- ・ CALL文
- ・ DISPLAY文
- ・ EXIT文
- ・ GO TO文
- ・ IF文
- ・ MOVE文
- ・ OPEN文
- ・ READ文
- ・ SET文
- ・ WRITE文

翻訳、リンク、実行方法

- ・ cobolコマンドを利用する場合

```
$ cobol -M -c -WC, "NOALPHAL" sample4. cob
最大重大度コードは I で, 翻訳したプログラム数は 1 本です.
$ cobol -c -WC, "NOALPHAL" print. cob
最大重大度コードは I で, 翻訳したプログラム数は 1 本です.
$ cobol -o sample4 sample4.o print.o
$ ./sample4
$ cat work | more
0123 定規                                0200
0456 ボールペン                          0100
0789 分度器                              0050
0812 カッター                            0250
0823 はさみ                              0450
0834 パンチ                              0500
0845 蛍光ペン                            0100
0856 シャープペン                        0200
0867 消しゴム                            0100
0878 修正ペン                            0300
--継続--
```

- ・ Makefileを利用する場合

```
$ make

$ ./sample4
$ cat work | more
0123 定規                                0200
```

0456 ボールペン	0100
0789 分度器	0050
0812 カッター	0250
0823 はさみ	0450
0834 パンチ	0500
0845 蛍光ペン	0100
0856 シャープペン	0200
0867 消しゴム	0100
0878 修正ペン	0300
--継続--	

6.5 コマンド行引数の受取り方(sample05)

ここでは、この製品で提供されているサンプルプログラム-sample05-について説明します。

sample05では、コマンド行引数の操作機能を使って、COBOLプログラムを実行するときに指定された引数を受け取るプログラムの例を示します。

コマンド行引数の操作機能の使い方は、“NetCOBOL ユーザーズガイド”の“コマンド行引数の取出し”を参照してください。

概要

開始年月日から終了年月日までの日数を求めます。開始年月日および終了年月日は、コマンドの引数として次の形式で指定します。

sample5	開始年月日	終了年月日
---------	-------	-------

開始年月日および終了年月日は、1900年1月1日～2172年12月31日をYYYYMMDDで指定します。西暦は4桁で指定します。

提供プログラム

- sample5.cob (COBOLソースプログラム)
- Makefile (メイクファイル)

使用しているCOBOLの機能

- コマンド行引数の取出し
- 内部プログラム

使用しているCOBOLの文

- ACCEPT文
- CALL文
- COMPUTE文
- DISPLAY文
- DIVIDE文
- EXIT文
- GO TO文
- IF文
- MOVE文
- PERFORM文

翻訳、リンク、実行方法

- cobolコマンドを利用する場合

```
$ cobol -M -o sample5 sample5.cob
最大重大度コードは I で、翻訳したプログラム数は 1 本です。
$ ./sample5 19710406 20000501
日数の差は +10617日です。
$
```

- Makefileを利用する場合

```
$ make

$ ./sample5 19710406 20000501
日数の差は +10617日です。
$
```

6.6 環境変数の操作(sample06)

ここでは、この製品で提供されているサンプルプログラム-sample06-について説明します。

sample06では、環境変数の操作機能を使って、プログラムの実行中に環境変数の値を変更するプログラムの例を示します。

環境変数の操作機能の使い方は、“NetCOBOL ユーザーズガイド”の“環境変数の操作機能”を参照してください。

概要

商品コード、商品名、および単価が格納されているマスタファイル中のデータを、商品コードの値によって二つのマスタファイルに分割します。分割方法と新規に作成する二つのマスタファイルのファイル名を以下に示します。

商品コードの値	ファイル名
先頭が“0”	master.a
先頭が“0”以外	master.b

提供プログラム

- sample6.cob(COBOLソースプログラム)
- Makefile(メイクファイル)
- master(マスタファイル)

使用しているCOBOLの機能

- 環境変数の操作機能
- 索引ファイル

使用しているCOBOLの文

- ACCEPT文
- CLOSE文
- DISPLAY文
- EXIT文
- GO TO文
- IF文
- MOVE文

- OPEN文
- READ文
- STRING文
- WRITE文

翻訳、リンク、実行方法

- cobolコマンドを利用する場合

```
$ cobol -M -o sample6 sample6.cob
最大重大度コードは I で、翻訳したプログラム数は 1 本です。
$ INFILE=master; export INFILE
$ OUTFILE=: export OUTFILE
$ ./sample6
$ ls mas*
master      master.a    master.b
$
```

- Makefileを利用する場合

```
$ make

$ INFILE=master; export INFILE
$ OUTFILE=: export OUTFILE
$ ./sample6
$ ls mas*
master      master.a    master.b
$
```

マスタファイルと同じディレクトリに次の2つのファイルが作成されます。

- master.a(商品コードの先頭が“0”の商品のデータを格納したマスタファイル)
- master.b(商品コードの先頭が“0”以外の商品のデータを格納したマスタファイル)

索引ファイル(master.a/master.b)が正しく作成されたことを確認する場合は、ファイルユーティリティを使用してください。ファイルユーティリティの表示コマンドでは、索引ファイルのレコードを表示することができます。使用方法の詳細は、“NetCOBOL ユーザーズガイド”の“ファイルユーティリティ”を参照してください。



参考

ソースプログラムのASSIGN句にファイル識別名を記述した場合、プログラムを実行する前にファイルを割り当てる必要があります。ファイルを割り当てるには、ファイル識別名を環境変数名として、使用するファイルの名前をその環境変数に設定します。このsampleでは、マスタファイルを割り当てる環境変数はINFILEで、新規に作成するファイルを割り当てる環境変数はOUTFILEです。ただし、新規に作成するファイルについては、プログラム中で環境変数の操作機能を使用してファイルの割当てを行っているため、プログラム実行前に設定している値は無視されます。



注意

すでにmaster.aおよびmaster.bというファイルが存在する場合、そのファイルの内容は失われます。

6.7 印刷ファイルを使ったプログラム(sample07)

ここでは、この製品で提供されているサンプルプログラム-sample07-について説明します。

sample07では、印刷ファイルを使って、標準入力から入力したデータを印刷装置に出力するプログラムの例を示します。

印刷ファイルの使い方の詳細は、“NetCOBOL ユーザーズガイド”の“行単位のデータを印刷する方法”を参照してください。

概要

標準入力先から英数字のデータを入力し、印刷装置に出力します。データの入力を終了するときには、“/END”に続けて36文字の空白を入力します。

提供プログラム

- sample7.cob(COBOLソースプログラム)
- Makefile(メイクファイル)
- COBOL.CBR(実行用初期化ファイル)

使用しているCOBOLの機能

- 印刷ファイル
- 小入出力機能

使用しているCOBOLの文

- ACCEPT文
- CLOSE文
- EXIT文
- GO TO文
- IF文
- OPEN文
- WRITE文

翻訳、リンク、実行方法

- cobolコマンドを利用する場合

```
$ cobol -M -o sample7 sample7.cob
最大重大度コードは I で、翻訳したプログラム数は 1 本です。
$ ./sample7
1234567890123456789012345678901234567890
abcde
*****sample7*****
/END

$
```

- Makefileを利用する場合

```
$ make

$ ./sample7
1234567890123456789012345678901234567890
abcde
*****sample7*****
/END

$
```

プログラムが終了すると、印刷装置にデータが出力されます。

```
1234567890123456789012345678901234567890
abcde*****sample7*****
```

… (注)

注：このsampleプログラムでは、1回のACCEPT文の実行はデータを40文字入力すると終了となります。したがって、2回目の入力データと3回目の入力データを合わせたデータがプログラムでの2回目のACCEPT文の実行でのデータの入力となります。

6.8 印刷ファイルを使ったプログラム(応用編)(sample08)

ここでは、この製品で提供されているサンプルプログラム-sample08-について説明します。

sample08では、I制御レコード、CHARACTER TYPE句およびPRINTING POSITION句を使用して、さまざまな指定を行うFORMAT句なし印刷ファイルのプログラムの例を示します。

FORMAT句なし印刷ファイルの使い方の詳細は、“NetCOBOL ユーザーズガイド”の“行単位のデータを印刷する方法”および“7.3 フォームオーバーレイおよびFCBを使う方法”を参照してください。

概要

FORMAT句なし印刷ファイルを使用して帳票印刷を行う場合、主に利用される機能を想定し、以下の項目について印刷デモを行います。

FCBを使用した6LPI、8LPIでの帳票印刷

FCBを利用した任意の行間隔(6/8LPI)で帳票印刷を行うことを想定し、I制御レコードによるFCB(LPI)の切り替えを行います。ソースプログラムには、CHARACTER TYPE句やPRINTING POSITION句を記述して、行間隔(LPI)や文字間隔(CPI)などの行・桁を意識して帳票の体裁を整えます。

以下の帳票印刷を行います。

- A4用紙を横向きに使用し、1ページすべての行間隔を6LPIとした場合の帳票をイメージし、6LPI/10CPIフォーマットのスペーシングチャート形式のフォームオーバーレイと重畳印刷します。
- A4用紙を横向きに使用し、1ページすべての行間隔を8LPIとした場合の帳票をイメージし、8LPI/10CPIフォーマットのスペーシングチャート形式のフォームオーバーレイと重畳印刷します。

CHARACTER TYPE句で指定する各種文字属性での印刷

I制御レコードを使用し、用紙サイズをA4/横向きからB4/横向きに変更し、これにあわせてFCBもA4/横向き用からB4/横向き用に変更します。

以下の各種文字属性の印字サンプルを印刷装置に出力します。

1. 文字サイズ

1文字ずつ3ポ、7.2ポ、9ポ、12ポ、18ポ、24ポ、36ポ、50ポ、72ポ、100ポ、200ポ、300ポの文字サイズを印字します。



参考

ここでは、文字ピッチ指定を省略することにより、文字サイズに合わせた最適な文字ピッチをCOBOLランタイムシステムに自動算出させます。

2. 文字ピッチ

文字ピッチ1CPIで1文字、2CPIで2文字、3CPIで3文字、5CPIで5文字、6CPIで6文字、7.5CPIで15文字、20CPIで20文字、24CPIで24文字指定します。



参考

ここでは、文字サイズ指定を省略することにより、文字ピッチに合わせた最適な文字サイズをCOBOLランタイムシステムに自動算出させます。

3. 文字書体

ゴシック、ゴシック半角(文字形態半角)、明朝、明朝半角(文字形態半角)を10文字ずつ二回繰り返し印字します。

4. 文字回転

縦書き(反時計回りに90度回転)、横書きを10文字ずつ繰り返し印字します。

5. 文字形態

全角、半角、全角平体、半角平体、全角長体、半角長体、全角倍角、半角倍角の文字形態指定を9文字ずつ印刷します。

6. 上記5つの文字属性を組み合わせた印刷を行います。

提供プログラム

- sample8.cob (COBOLソースプログラム)
- kol5/A4L6 (フォームオーバーレイパターン)
- kol5/A4L8 (フォームオーバーレイパターン)
- kol5/B4OV (フォームオーバーレイパターン)
- fcb/A4L6 (FCB)
- fcb/A4L8 (FCB)
- fcb/LPI6 (FCB)
- infofile/PRTINFO.UVPI (印刷情報ファイル)
- sample8.sh (実行用のシェル)
- Makefile (メイクファイル)

使用しているCOBOLの機能

- 印刷ファイル
- 小入出力機能

使用しているCOBOLの文

- ADD文
- CLOSE文
- DISPLAY文
- IF文
- MOVE文
- OPEN文
- PERFORM文
- STOP文
- WRITE文

翻訳、リンク

サンプル格納ディレクトリが“/home/samples/sample08”であると仮定します。

```
$ cd /home/samples/sample08
$ COBPATH=/home/samples/sample08; export COBPATH
$ make
...
```

実行方法

以下に、sample8の実行例を示します。

```
$ cd /home/samples/sample08
$ COBPATH=/home/samples/sample08; export COBPATH
$ ./sample8.sh uvpi
```

プログラムが終了すると、印刷装置にデータが出力されます。

環境変数COBPATHには、sample08の資源を格納したディレクトリを絶対パスで指定してください。



SELinuxを有効にした環境でサンプルプログラムを実行する場合は、印刷資源(フォームオーバーレイ、FCBファイル)の配置に注意が必要です。詳細は、“PrintWalker/LXE説明書(機能編)”を参照してください。

6.9 FORMAT句付き印刷ファイルを使ったプログラム(sample09)

ここでは、この製品で提供されているサンプルプログラム-sample09について説明します。

sample09では、FORMAT句付き印刷ファイルを使って、集計表を印刷装置に出力するプログラムの例を示します。

FORMAT句付き印刷ファイルの使い方は、“NetCOBOL ユーザーズガイド”の“帳票定義体を使う印刷ファイルの使い方”を参照してください。

なお、このプログラムを実行するためには、MeFtが必要です。

概要

商品コード、商品名および単価が格納されているマスタファイル(master)と受注日、数量および売上げ金額が格納されている売上げファイル(sales)を入力して、売上集計表を印刷装置に出力します。

使用している帳票定義体

売上集計表(SYUUKEI.pmd)

形式	集計表形式	
用紙サイズ	A4	
用紙方向	縦	
行ピッチ	1/6インチ	
パーティション	PH(ページ頭書き)	[固定パーティション、印刷開始位置:0インチ(1行目)、縦幅:1インチ(6行)]
	CH1(制御頭書き)	[浮動パーティション、縦幅:0.83インチ(5行)]
	DE(明細)	[浮動パーティション、縦幅:0.33インチ(2行)]
	CF1(制御脚書き)	[浮動パーティション、縦幅:0.83インチ(5行)]
	CF2(制御脚書き)	[浮動パーティション、縦幅:0.67インチ(4行)]
	PF(ページ脚書き)	[固定パーティション、印刷開始位置:10.48インチ(63行目)、縦幅:0.49インチ(3行)]

帳票定義体の内容を確認する場合には、Windows上のPowerFORMを使用してください。

プログラムを作成するうえでのポイント

- PHおよびPFは、固定パーティション(固定の印刷位置情報を持っている)なので、パーティションを出力すると、必ず、パーティションに定義されている印刷開始位置に出力されます。
- CH1、DE、CF1およびCF2は、浮動パーティション(固定の印刷位置情報を持たない)なので、自由な位置に出力することができる反面、パーティション出力時に印刷位置を制御する必要があります。
- 各パーティションに定義された出力項目は、翻訳時にCOPY文で帳票定義体からレコードに展開されます。このとき、定義した出力項目の項目名がデータ名になります。

提供プログラム

- sample9.cob(COBOLソースプログラム)
- SYUUKKEI.pmd(帳票定義体)
- SYOHINM.cbl(登録集原文)
- URIAGE.cbl(登録集原文)
- mefprc(プリンタ情報ファイル)
- Makefile(メイクファイル)
- sales(索引ファイル)
- master(マスタファイル)
- COBOL.CBR(実行用初期化ファイル)

使用しているCOBOLの機能

- FORMAT句付き印刷ファイル
- 索引ファイル(参照)
- 登録集の取込み
- 小入出力機能

使用しているCOBOLの文

- OPEN文
- READ文
- WRITE文
- START文
- CLOSE文
- PERFORM文
- DISPLAY文
- IF文
- MOVE文
- SET文
- GO TO文
- EXIT文
- COPY文
- ADD文

プログラムを実行する前に

- MeFtのセットアップを行い、使用できる状態にしておいてください。
- プリンタ情報ファイルの設定を使用する環境にあわせて修正してください。
プリンタ情報ファイル(mefprc)の下線部の情報を、エディタを使用して変更しておきます。

<pre>PRTNAME printer-name MEDDIR /opt/FJSVcbl64/samples/ja_JP.UTF-8/sample09</pre>
--

ー PRTNAME : 帳票を出力する印刷装置名

— MEDDIR: 帳票定義体(SYUUKEL.pmd)を格納したパス名

プリンタ情報ファイルに記述する内容の詳細については“MeFtユーザーズガイド”を参照してください。

翻訳、リンク、実行方法

- cobolコマンドを利用する場合

```
$ cobol -M -o sample9 sample9.cob
最大重大度コードは I で、翻訳したプログラム数は 1 本です.
$ ./sample9
$
```

- Makefileを利用する場合

```
$ make
$ ./sample9
$
```

6.10 オブジェクト指向プログラム(初級編)(sample10)

ここでは、この製品で提供されているサンプルプログラム-sample10-について説明します。

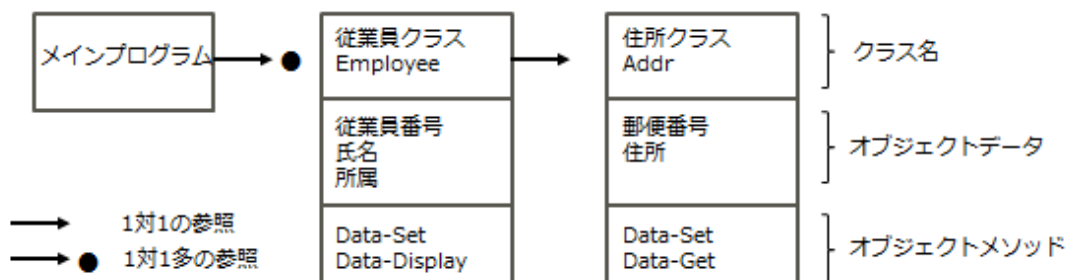
sample10では、オブジェクト指向プログラミング機能を使ったプログラムの例を示します。このプログラムでは、カプセル化、オブジェクトの生成、メソッド呼出しといった、オブジェクト指向の基本的な機能だけを使用しています。

概要

最初に従業員オブジェクトを三つ生成しています。“NEW”メソッドでオブジェクトを生成した後、“Data-Set”を呼び出してデータを設定しています。それぞれの従業員オブジェクトはすべて同じ形をしています。しかし、持っているデータ(従業員番号、氏名、所属、住所情報)は異なります。また、住所情報もオブジェクトであり、郵便番号と住所を持っています。

画面から従業員番号を入力すると、当該従業員オブジェクトに対して“Data-Display”メソッドを呼び出し、そのオブジェクトが持っている従業員情報を画面に表示します。このとき、従業員オブジェクトは、住所の情報を得るために、従業員オブジェクトが指している住所オブジェクトに対し、“Data-Get”メソッドを呼び出します。

従業員オブジェクトは、三つのデータと住所オブジェクトから構成されています。しかし、これを使う側(この場合はメインプログラム)はオブジェクトの構造を知っている必要はありません。“Data-Set”と“Data-Display”の二つのメソッドだけを知っていれば十分です。つまり、データとアクセス手段を一つにまとめる(カプセル化)ことにより、オブジェクト内部の情報を完全に隠蔽しているわけです。



使用しているCOBOLの機能

- オブジェクト指向プログラミング機能
 - クラスの定義(カプセル化)
 - オブジェクトの生成
 - メソッド呼出し

使用しているオブジェクト指向の文/段落/定義

- INVOKE文、SET文
- リボジトリ段落
- クラス定義、オブジェクト定義、メソッド定義

提供プログラム

- main.cob (COBOLソースプログラム)
- member.cob (COBOLソースプログラム)
- address.cob (COBOLソースプログラム)
- Makefile (メイクファイル)

翻訳、リンク、実行方法

```
$ make
cobol -c -WC,"CREATE(REP)" -dr. address.cob
最大重大度コードは 1 で、翻訳したプログラム数は 1 本です。
cobol -c -WC,"CREATE(REP)" -dr. member.cob
最大重大度コードは 1 で、翻訳したプログラム数は 1 本です。
cobol -dr. -M -c main.cob
最大重大度コードは 1 で、翻訳したプログラム数は 1 本です。
cobol -dr. -c address.cob
最大重大度コードは 1 で、翻訳したプログラム数は 1 本です。
cobol -dr. -c member.cob
最大重大度コードは 1 で、翻訳したプログラム数は 1 本です。
cobol -o sample10 main.o address.o member.o
$ ./sample10
従業員番号を入力して下さい (1 or 2 or 3)
1

NO. 氏名-----所属-----〒-----住所-----
0001 富士通一郎      営業第一課      410-0007 静岡県沼津市北沢田 3 2 8-1 1

終了しますか? (Y/N)
Y

$
```

6.11 データベース機能を使ったプログラム(sample16)

ここでは、本製品で提供するサンプルプログラム-sample16について説明します。sample16では、データベース(SQL)機能を使ってデータベースからデータを取り出しホスト変数に格納する例を示します。

データベースはサーバ上に存在し、クライアント側からこれにアクセスします。データベースのアクセスは、ODBCドライバを経由して行います。

ODBCドライバを使用するデータベースアクセスについては、“NetCOBOL ユーザーズガイド”の“リモートデータベースアクセス”を参照してください。

このプログラムを動作させるためには、以下の製品が必要です。

クライアント側

- ODBCドライバマネージャ
- ODBCドライバ
- ODBCドライバの必要とする製品

サーバ側

- データベース

- ー データベースにODBCでアクセスするために必要な製品

sample16では、次に示すCOBOLの文を使っています。

- ・ DISPLAY文
- ・ GO TO文
- ・ IF文
- ・ PERFORM文
- ・ ACCEPT文

sample16では、次に示す埋込みSQL文を使っています。

- ・ 埋込み例外宣言
- ・ CONNECT文
- ・ カーソル宣言
- ・ OPEN文
- ・ FETCH文
- ・ CLOSE文
- ・ ROLLBACK文
- ・ DISCONNECT文

機能

サーバのデータベースにアクセスし、データベース上の表“STOCK”に格納されている全データをディスプレイ装置に表示します。データをすべて参照し終わると、データベースとの接続を切断します。

プログラムを実行する前に

ODBCドライバを経由してサーバのデータベースへアクセスできる環境を構築しておいてください。SERVERNAME1というサーバ名で接続するサーバを設定し、そのサーバのデータベース上に“STOCK”という名前で表を作成しておいてください。STOCK表は、以下の形式で作成してください。

列の名前	列の属性
GNO	2進整数、4桁
GOODS	固定長文字、20バイト
QOH	2進整数、9桁
WHNO	2進整数、4桁

STOCK表に格納しておくデータは任意です。以下に例を示します。

GNO	GOODS	QOH	WHNO
110	TELEVISION	85	2
111	TELEVISION	90	2
123	REFRIGERATOR	60	1
124	REFRIGERATOR	75	1
137	RADIO	150	2
138	RADIO	200	2
140	CASSETTE DECK	120	2
141	CASSETTE DECK	80	2

GNO	GOODS	QOH	WHNO
200	AIR CONDITIONER	04	1
201	AIR CONDITIONER	15	1
212	TELEVISION	0	2
215	VIDEO	05	2
226	REFRIGERATOR	08	1
227	REFRIGERATOR	15	1
240	CASSETTE DECK	25	2
243	CASSETTE DECK	14	2
351	CASSETTE TAPE	2500	2
380	SHAVER	870	3
390	DRIER	540	3

ODBC情報テンプレート(/opt/FJSVcbl64/config/template/ja_JP.UTF-8/odbcinf)を利用してODBC情報ファイルを作成してください。ここではDBMSACS.INFとします。

実行方法

以下に、sample16の実行例を示します。

```
$ cobol -M -o sample16 sample16.cob
最大重大度コードは 1 で、翻訳したプログラム数は 1 本です。
$ ODBC_INF= DBMSACS.INF; export ODBC_INF
$ ./sample16
$
```

以下の要求がACCEPT文によって、行われます。

ユーザID/パスワードを入力して下さい(ID/PASSWD):

要求に対し、ユーザIDとパスワードを斜線"/"で区切って入力してください。入力文字が65バイト以下の場合、65バイトになるように空白を入力して下さい。

実行結果

表から取り出されるデータは、以下のように表示されます。

```
$ sample16
:
16件目のデータ:
  製品番号 = +0243
  製品名   = CASSETTE DECK
  在庫数量 = +000000014
  倉庫番号 = +0002
17件目のデータ:
  製品番号 = +0351
  製品名   = CASSETTE TAPE
  在庫数量 = +000002500
  倉庫番号 = +0002
18件目のデータ:
  製品番号 = +0380
  製品名   = SHAVER
  在庫数量 = +000000870
  倉庫番号 = +0003
19件目のデータ:
  製品番号 = +0390
```

製品名	=	DRIER
在庫数量	=	+000000540
倉庫番号	=	+0003
終了します		
\$		

6.12 データベース機能を使ったプログラム(応用編)(sample17)

ここでは、本製品で提供するサンプルプログラム-sample17-について説明します。sample17では、データベース(SQL)機能のより進んだ使い方として、データベースの複数の行を1つの埋込みSQL文で操作する例を示します。

データベースはサーバ上に存在し、クライアント側からこれにアクセスします。データベースのアクセスは、ODBCドライバを経由して行います。

ODBCドライバを使用するデータベースアクセスについては、“NetCOBOL ユーザーズガイド”の“リモートデータベースアクセス”を参照してください。

このプログラムを動作させるためには、以下の製品が必要です。

クライアント側

- ODBCドライバマネージャ
- ODBCドライバ
- ODBCドライバの必要とする製品

サーバ側

- データベース
- データベースにODBCでアクセスするために必要な製品

sample17では、次に示すCOBOLの文を使っています。

- CALL文
- ACCEPT文
- DISPLAY文
- GO TO文
- IF文
- PERFORM文

sample17では、次に示す埋込みSQL文を使っています。

- 埋込み例外宣言
- CONNECT文
- カーソル宣言
- OPEN文
- FETCH文
- SELECT文
- DELETE文
- UPDATE文
- CLOSE文
- COMMIT文
- ROLLBACK文
- DISCONNECT文

- ・ 複数行指定ホスト変数
- ・ 表指定ホスト変数

機能

sample16と同じデータベースにアクセスし、次の操作を行ってからデータベースとの接続を切断します。

- ・ データベース全データの表示
- ・ GOODSの値が“TELEVISION”である行のGNOの値の取り出し
- ・ 取り出したGNOを持つ行のQOHの更新
- ・ GOODSの値が“RADIO”、“SHAVER”、“DRIVER”の行の削除
- ・ データベースの全データの再表示

プログラムを実行する前に

ODBCドライバを経由してサーバのデータベースへアクセスできる環境を構築しておいてください。SERVERNAME1というサーバ名で接続するサーバを設定し、そのサーバのデータベース上に“STOCK”という名前で表を作成しておいてください。

STOCK表は、以下の形式で作成してください。

列の名称	列の属性
GNO	2進整数、4桁
GOODS	固定長文字、20バイト
QOH	2進整数、9桁
WHNO	2進整数、4桁

STOCK表に次のデータを格納しておいてください。

GNO	GOODS	QOH	WHNO
110	TELEVISION	85	2
111	TELEVISION	90	2
123	REFRIGERATOR	60	1
124	REFRIGERATOR	75	1
137	RADIO	150	2
138	RADIO	200	2
140	CASSETTE DECK	120	2
141	CASSETTE DECK	80	2
200	AIR CONDITIONER	04	1
201	AIR CONDITIONER	15	1
212	TELEVISION	0	2
215	VIDEO	05	2
226	REFRIGERATOR	08	1
227	REFRIGERATOR	15	1
240	CASSETTE DECK	25	2
243	CASSETTE DECK	14	2
351	CASSETTE TAPE	2500	2
380	SHAVER	870	3

GNO	GOODS	QOH	WHNO
390	DRIER	540	3

ODBC情報テンプレート(/opt/FJSVcbl64/config/template/ja_JP.UTF-8/odbcinf)を利用してODBC情報ファイルを作成してください。ここではDBMSACS.INFとします。

実行方法

以下に、sample17の実行例を示します。

```
$ cobol -M -osample17 sample17.cob
最大重大度コードは 1 で、翻訳したプログラム数は 1 本です。
$ ODBC_INF= DBMSACS.INF; export ODBC_INF
$ ./sample17
```

以下の要求がACCEPT文によって、行われます。

ユーザID/パスワードを入力して下さい(ID/PASSWD)：

要求に対し、ユーザIDとパスワードを斜線"/"で区切って入力してください。入力文字が

65バイト以下の場合、65バイトになるように空白を入力して下さい。

実行結果

表から取り出されるデータは、以下のように表示されます。

接続に成功しました

処理前のテーブルの内容

01件目のデータ：

```
製品番号 = +0110
製品名   = TELEVISION
在庫数量 = +000000085
倉庫番号 = +0002
```

：

19件目のデータ：

```
製品番号 = +0390
製品名   = DRIER
在庫数量 = +000000540
倉庫番号 = +0003
```

全データ件数は19件です

製品名がTELEVISIONの行の製品番号を取り出します。

以下の製品の在庫数量を10ずつ減少させます。

```
TELEVISION -> +0110
TELEVISION -> +0110
TELEVISION -> +0212
```

製品名が'RADIO'、'SHAVER'、'DRIER'の行を削除します

処理後のテーブルの内容

01件目のデータ：

```
製品番号 = +0110
製品名   = TELEVISION
在庫数量 = +000000065
倉庫番号 = +0002
```

：

15件目のデータ :

製品番号 = +0351

製品名 = CASSETTE TAPE

在庫数量 = +000002500

倉庫番号 = +0002

全データ件数は15件です

終了します

\$

6.13 COBOLファイルアクセスルーチンのサンプル

6.13.1 行順ファイルの読み込み

概要

このサンプルプログラムは、指定したファイルを行順ファイルとしてINPUTモードでオープンし、読み込んだレコードの内容を表示します。

提供プログラム

- cobfa01.c (Cソースプログラム)
- Makefile(メイクファイル)

使用しているCOBOLファイルアクセスルーチンの関数

- cobfa_open()関数
- cobfa_rdnnext()関数
- cobfa_stat()関数
- cobfa_errno()関数
- cobfa_reclen()関数
- cobfa_close()関数

プログラムの翻訳とリンク

Makefileの“CDIR=”と書かれている行の右側の内容が、Cコンパイラをインストールしたディレクトリ名となるように修正してください。また“COBDIR=”と書かれている行の右側の内容が、COBOLコンパイラをインストールしたディレクトリ名となるように修正してください。

Makefileを修正した後、以下のコマンドを入力します。

```
$ make
```

プログラムの実行

適当なテキストファイルをコマンドライン引数にしてプログラムを実行します。ここではcobfa01自身のソースプログラムを入力します。

```
$ ./cobfa01 cobfa01.c
```

6.13.2 行順ファイルの読み込みと索引ファイルの書出し

概要

このサンプルプログラムは、特定の行順ファイル(cobfa02.txt)をINPUTモードでオープンし、そのレコードの内容を索引ファイル(cobfa02.idx)のレコードとして書き出します。

最後に、その索引ファイルをINPUTモードでオープンし、主キーの順で画面に表示します。

提供プログラム

- cobfa02.c (Cソースプログラム)
- cobfa02.txt (入力用行順ファイル)
- Makefile(メイクファイル)

使用しているCOBOLファイルアクセスルーチンの関数

- cobfa_open()関数
- cobfa_rdnnext()関数
- cobfa_wrkey()関数
- cobfa_stkey()関数
- cobfa_close()関数

プログラムの翻訳とリンク

Makefileの“CDIR=”と書かれている行の右側の内容が、Cコンパイラをインストールしたディレクトリ名となるように修正してください。また“COBDIR=”と書かれている行の右側の内容が、COBOLコンパイラをインストールしたディレクトリ名となるように修正してください。

Makefileを修正した後、以下のコマンドを入力します。

```
$ make
```

プログラムの実行

コマンドライン引数を付けずに実行します。

```
$ ./cobfa02
```

6.13.3 索引ファイルの情報の取得

概要

このサンプルプログラムは、指定したファイルを索引ファイルとしてINPUTモードでオープンし、ファイル自体の属性と、レコードキーの各構成を表示します。

提供プログラム

- cobfa03.c (Cソースプログラム)
- Makefile(メイクファイル)

使用しているCOBOLファイルアクセスルーチンの関数

- cobfa_open()関数
- cobfa_indexinfo()関数
- cobfa_stat()関数
- cobfa_errno()関数
- cobfa_close()関数

プログラムの翻訳とリンク

Makefileの“CDIR=”と書かれている行の右側の内容が、Cコンパイラをインストールしたディレクトリ名となるように修正してください。また“COBDIR=”と書かれている行の右側の内容が、COBOLコンパイラをインストールしたディレクトリ名となるように修正してください。

Makefileを修正した後、以下のコマンドを入力します。

```
$ make
```

プログラムの実行

適当な索引ファイルをコマンドライン引数にしてプログラムを実行します。ここではcobfa02を実行して生成した索引ファイルを指定します。

```
$ ./cobfa03 ../cobfa02/cobfa02.idx
```

索引

[数字]	[た]
10進項目.....67	注意事項.....68
2進項目.....67	帳票印刷.....72
[C]	テキストファイル.....73
COBOLパースペクティブ.....10	デバッグパースペクティブ.....10
COBOLプログラム間の呼出し.....73	データベース機能.....84,87
[E]	登録集の取込み.....72,74
Eclipse.....9	[は]
[I]	パースペクティブ.....10
INVOKE文.....84	表示ファイル機能.....71
[M]	標準出力.....69
MeFt.....71	標準入出力を使ったデータ処理.....69
[N]	標準入力.....69,77
NetCOBOL Studio.....9	浮動小数点項目.....68
[S]	プログラム間結合.....66
SET文.....84	プログラム間連絡機能.....74
svdrdsコマンド.....18	プロジェクト.....9
[あ]	ポート番号.....14
印刷可能文字.....73	[ま]
印刷装置.....77	メソッド定義.....84
印刷ファイル.....74,77	メソッド呼出し.....83
印刷ファイルを使ったプログラム.....77,79,81	[ら]
オブジェクト指向プログラミング機能.....83	リポジトリ段落.....84
オブジェクト指向プログラム.....83	[わ]
オブジェクト定義.....84	ワークスペース.....9
オブジェクトの生成.....83	ワークベンチ.....9
[か]	
カプセル化.....83	
画面入出力.....72	
簡易入出力.....69	
環境変数の操作.....76	
環境変数の操作機能.....76	
行順ファイル.....70,74	
行順ファイルと索引ファイルの操作.....70	
行順ファイルの読み込み.....90	
行順ファイルの読み込みと索引ファイルの書出し.....90	
クラス定義.....84	
コマンド行引数の受け取り方.....75	
コマンド行引数の操作機能.....75	
コマンド行引数の取出し.....75	
[さ]	
索引ファイル.....70,74	
索引ファイルの情報の取得.....91	
サブルーチン.....73	
実行時パラメタの受渡し.....74	
小入出力機能.....72,74,78	
数字項目の標準規則.....67	