Visual COBOL チュートリアル

COBOL 開発： Linux/UNIX 版 リモート開発編

1. 目的

本チュートリアルは、Visual COBOL の Linux/UNIX が提供するリモート開発機能を学ぶためのチュートリアルです。リモート開発を利用することで、以下のメリットを享受できるようになります。

- 高機能なオープンソースの IDE（統合開発環境）として広く普及する Eclipse 上で Linux/UNIX 環境をターゲットとしたアプリケーション開発が行えるため、Linux/UNIX 環境における開発効率の向上

2. 前提

- 本チュートリアルで使用したマシン OS ： Windows Server 2022 Standard Edition, Red Hat Enterprise Linux 8.5
- Windows 上に Visual COBOL 9.0J for Eclipse がインストール済みであること
- Visual COBOL 9.0J Development Hub がインストール済みであること

本チュートリアルの実施前に、「Visual COBOL for Eclipse チュートリアル」を実施してください。

内容

3. チュートリアル手順の概要

3.1. 環境のセットアップ

Visual COBOL の Linux/UNIX 版の開発ライセンスは Windows にインストールして利用する Visual COBOL for Eclipse と Linux/UNIX 環境にインストールする Visual COBOL Development Hub がセットになったライセンスです。リモート開発を行うためには、両製品ともにセットアップを行う必要があります。

3.1.1. Visual COBOL for Eclipse のセットアップ

未実施の場合、「Visual COBOL for Eclipse チュートリアル」の内容に従ってセットアップ、および、チュートリアル内容を先に行ってください。

3.1.2. Visual COBOL Development Hub のセットアップ

1) 入手したインストールプログラムをターゲットの OS へファイル転送します。
2) リリースノートを確認し、インストール要件を満たしていることを確認します。
3) 転送したインストールプログラムを解凍します。
4) 管理者権限を持ったユーザーへ切り替えます。
5) 解凍したインストーラへ実行権限を与えます[1]。

```
# chmod u+x setup_visualcobol_devhub_9.0_redhat_x86_64
```

6) インストール処理を開始します[2]。

```
# ./setup_visualcobol_devhub_9.0_redhat_x86_64 -installlocation=/opt/microfocus/VC90
    -=-================================================-=-
                    Micro Focus Product - Product Extractor
                                www.microfocus.com
                                    ～中略～
   製品をインストールする前に「使用許諾契約」のコピーが必要な場合は、
   同意しないで、次のコマンドでインストーラを再度実行してください ：

       ./setup_visualcobol_devhub_9.0_redhat_x86_64 -EULA

使用許諾契約の条件に同意しますか? (y/n): y

   使用許諾契約（EULA）は製品ディレクトリの次のファイルで確認できます:
       /opt/microfocus/VC90/etc/EULA_VCED_v9_0_jp.htm
```

使用許諾契約の条件を確認し、問題なければ「y」を入力します。

[1] インストーラのファイル名は、「setup_visualcobol_devhub_9.0_<プラットフォーム名>」の形式で構成されており、x86_64 Red Hat 版以外の製品を利用される場合はこの部分が異なるため、注意してください。実行時はファイル名に合わせて適切な名前に置き換えてください。

[2] デフォルトのインストールディレクトリは「/opt/microfocus/VisualCOBOL」です。本例では「-installlocation=/opt/microfocus/VC90」を指定し、インストールディレクトリを変更しています

```
Micro Focus Visual COBOL Development Hub 9.0 の SOA サポートを構成するには、
    $COBDIR/bin/casperm.sh を実行してください。
  -=-================================================-=-
                Micro Focus Visual COBOL Development Hub 9.0
                  インストールが完了しました。

使用許諾契約（EULA）は製品ディレクトリの次のファイルで確認できます:
     /opt/microfocus/VC80/etc/EULA_VCED_v9_0.htm

 -=-================================================-=-

      このバージョンの次の製品を使用するには :
                Micro Focus Visual COBOL Development Hub 9.0

      環境を設定するため、"cobsetenv" を実行してください。

              . /opt/microfocus/VC90/bin/cobsetenv

 -=-================================================-=-
```

3.2. Development Hub のインストール確認

Visual COBOL Development Hub は本書で紹介するリモート開発機能に加えて従来の COBOL 製品が提供するコマンドラインインターフェース機能も引き継いでいます。本章では前章でインストールした Visual COBOL Development Hub が正しくインストールされたことをこのコマンドラインインターフェースを使ったコンパイル及びテスト実行作業を通じて確認します。

1) ライセンスが未投入の場合は、インストールマニュアルに従い、ライセンスを適用します。

2) 一般ユーザーに戻ります。

3) Visual COBOL の利用に必要な環境変数を整えます。

Visual COBOL Development Hub をインストールすると Visual COBOL の利用に最低限必要な環境変数をセットアップするスクリプトが

＜インストールディレクトリ＞/bin/cobsetenv

に用意されます。本ステップではこのセットアップスクリプトを実行して環境変数設定をします。

```
$ . /opt/microfocus/VC90/bin/cobsetenv
COBDIR set to /opt/microfocus/VC90
$
```

このスクリプトにより設定される主な環境変数を下記に記します。

> COBDIR	製品のベースディレクトリ(インストールディレクトリ)
> PATH	$COBDIR/bin
> ライブラリ探索パス[3]	$COBDIR/lib

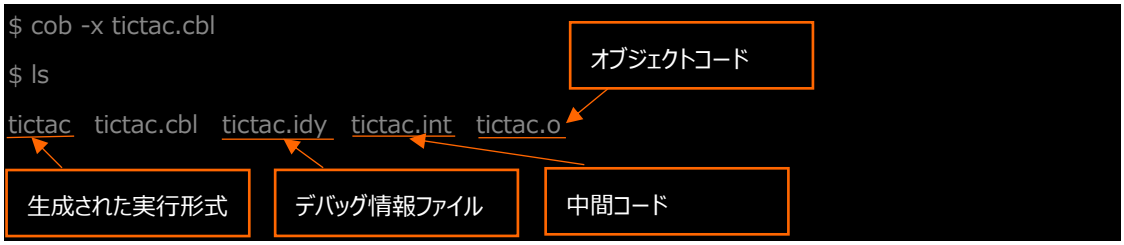
4) 製品同梱サンプルをコピーします。

Visual COBOL Development Hub をインストールすると $COBDIR/demo ディレクトリ配下にサンプルプログラム及びビルドスクリプトがカテゴリ分けされて配置されます。ここでは、このサンプル中における簡単なコンソールアプリケーションプログラムをワークディレクトリにコピーします。

```
$ cp -p $COBDIR/demo/cobol/tictac/*.cbl ./
$ ls
tictac.cbl
```

5) コピーしたプログラムを実行形式にコンパイルします。

Visual COBOL は COBOL プログラムを実行形式、ライブラリファイル、呼び出し可能な共有オブジェクト、動的ロードモジュール等、目的に応じて適切な形式にビルドする機能を持っています。ここでは、コピーしたサンプルプログラムを実行形式ファイルへシングルステップでビルドします。下記のコマンド実行結果からもわかるように、この 1 つのコマンドにより、中間コード、オブジェクトコードの生成並びに実行形式へのリンクが処理されていることがわかります。



6) ビルドしたアプリケーションをテスト実行します。

[3] LD_LIBARY_PATH, LIBPATH, SHLIB_PATH 等、プラットフォームによって環境変数名は異なります。

Visual COBOL Development Hub にはテスト実行機能が装備されており、コンパイル・ビルドしたモジュールを同環境上でテスト実行することが可能です。ここではこの機能を使って、生成した実行形式のプログラムをテスト実行してみます。tictac は○×ゲームのロジックを COBOL で組み上げたものとなります。プロンプトに従って○×ゲームを進めてみてください。

【実行イメージ】

① 実行形式を実行

```
$ ./tictac
To select a square type a number between 1 and 9
Shall I start ?
```

② 先攻／後攻を選択、本例では player が先攻となるよう選択

```
Shall I start ? n
```

③ ゲーム画面に切り替わるので、フィールドを選択してゲームを実行

```
To select a square type a number between 1 and 9
Please select an empty square 0
                          7|      8|      9
                           |       |
                           |       |
                       -------+-------+-------
                          4|      5|      6
                           |       |
                           |       |
                       -------+-------+-------
                          1|      2|      3
                           |       |
                           |       |
```

7) ゲーム終了後、アプリケーションの終了を選択

```
To select a square type a number between 1 and 9
stalemate
Play again ? n
```

3.3. リモートサーバーを起動

リモート開発は実際の操作対象が Linux/UNIX 側にあるにもかかわらず Windows 上の Eclipse にてあたかもローカルのリソースをコーディング編集やデバッグするかのようにして操作させることを可能にする技術です。このリモート開発を実行するにあたり、Linux/UNIX 側では Windows からの操作要求を受け付けるためのリモートサーバーを起動する必要があります。

Linux/UNIX と Windows の間の接続には、Eclipse が提供する RSE(Remote System Explorer) フレームワーク、もしくは SAMBA や NFS のようなネットワークファイルシステムが利用できます。ここではパフォーマンスの観点で有利な RSE で接続してみます。この RSE に関してもプロジェクトのポリシー（管理者権限を持ったユーザーの利用制限、ファイヤウォール等）に応じて柔軟に対応できるよう、デーモンを使って接続を自動確立させる方法並びに SSH でマニュアル接続させる方法を用意しています。本章では、デーモンによる自動接続を使った方法を紹介します。

1) 管理者権限を持ったユーザーに切り替えます。
2) ユーザーロケールを SJIS に設定します。
3) Visual COBOL の利用に必要な環境変数を整えます[4]。

```
# . /opt/microfocus/VC90/bin/cobsetenv
COBDIR set to /opt/microfocus/VC90
```

4) デーモンを起動します[5]。

```
# $COBDIR/remotedev/startrdodaemon
Starting RSE daemon...
Daemon running on: localhost.localdomain, port: 4075
```

[4] OS によっては、su コマンドの実行時にライブラリ探索パスをクリアするものもあるようです。前章で設定した環境変数が正しく引き継げていれば本作業は不要です。

[5] デフォルトでは、4075 ポートはデーモンに、ランダムな 5 桁のポートについては各 Windows との通信用に割り当てます。これらのポートがファイヤウォール等により閉じている場合は、
$COBDIR/remotedev/startrdodaemon <デーモンのポート番号> <Windows との通信ポート範囲>
のような形式で任意のポートへ割り当てることも可能です。

3.4. COBOL リモートプロジェクトの作成

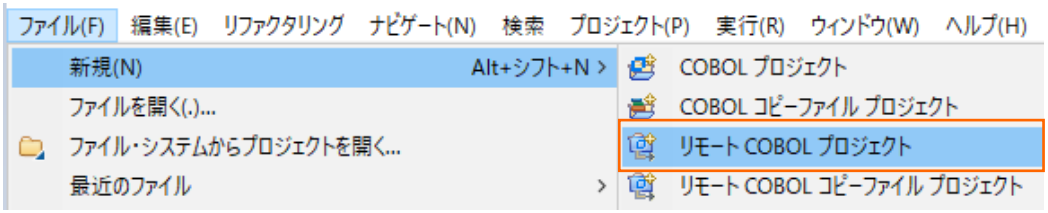
前章にて Linux/UNIX 環境側で Windows と通信するための準備作業が完了しました。ここからは、Windows 上にインストールされた Visual COBOL for Eclipse を使って Linux/UNIX 環境上に直接 COBOL アプリケーションをビルド生成してみます。アプリケーションのリソースは事前学習で利用した「Visual COBOL for Eclipse チュートリアル」で用意したものを利用します。

1) Visual COBOL for Eclipse を起動します。

ワークスペースの指定は特にありません。

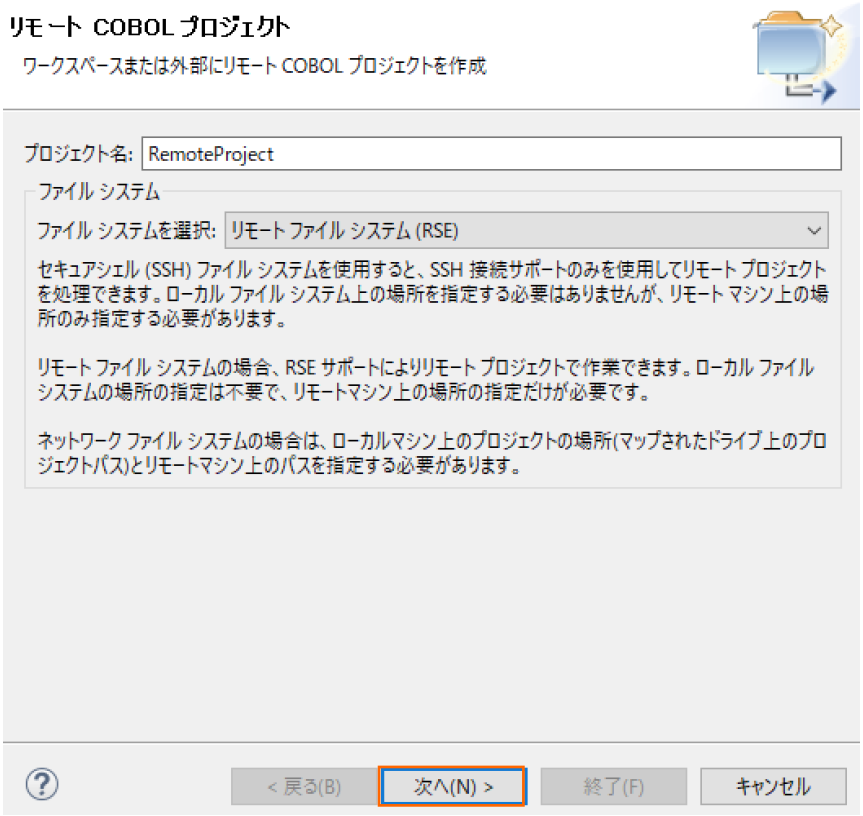
2) COBOL リモート プロジェクトを作成します。

① [ファイル(F)] メニューから [新規(N)] > [リモート COBOL プロジェクト] を選択します。

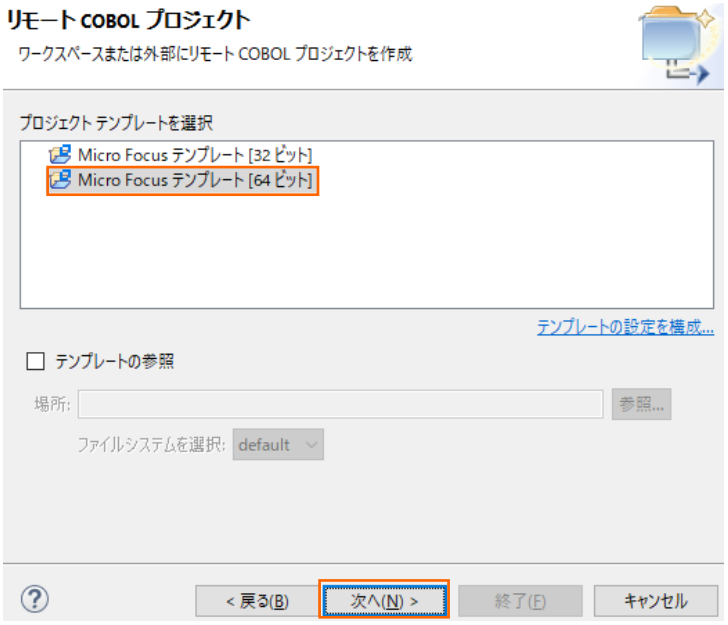


② [プロジェクト名] 欄に任意のプロジェクト名を指定し、[次へ(N)] ボタンをクリックします。

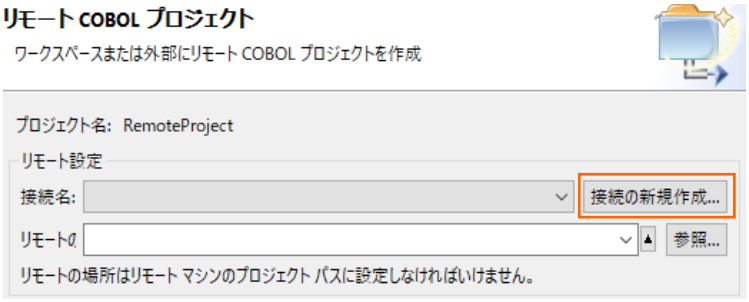
[ファイル システムを選択]は「リモートファイルシステム(RSE)」を選んでおきます。



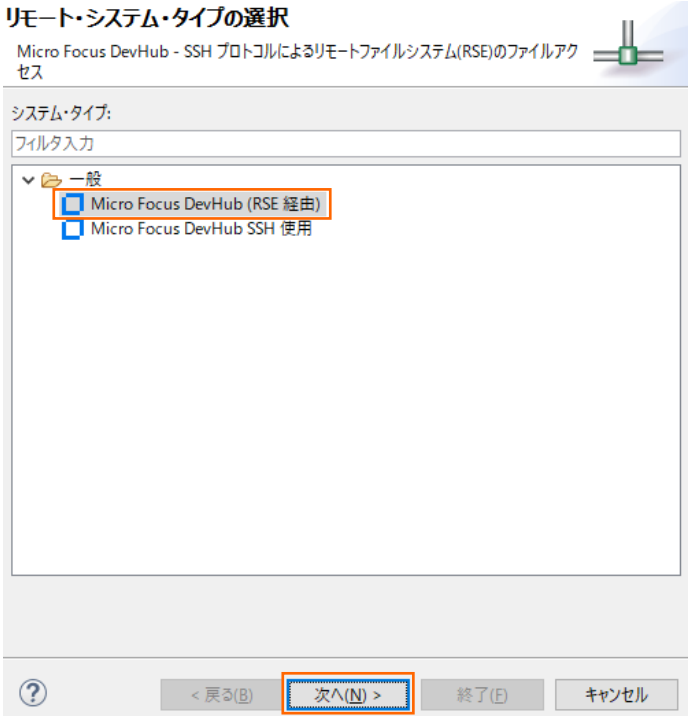
③ プロジェクトテンプレートは [Micro Focus テンプレート[64 ビット]] を選択し [次へ(N)] ボタンをクリックします。



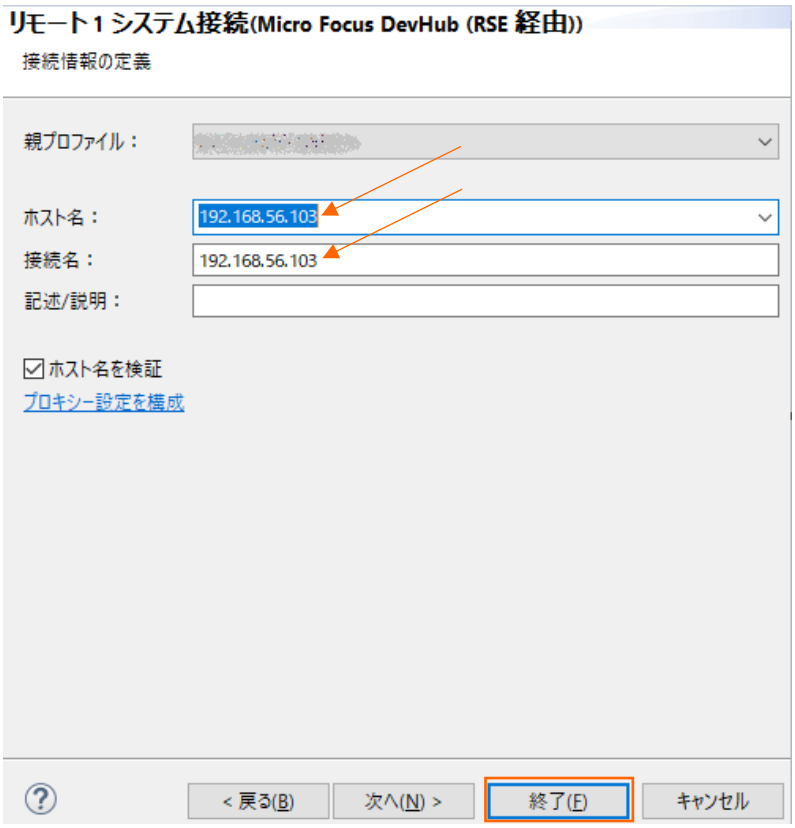
④ [接続の新規作成] ボタンをクリックします。



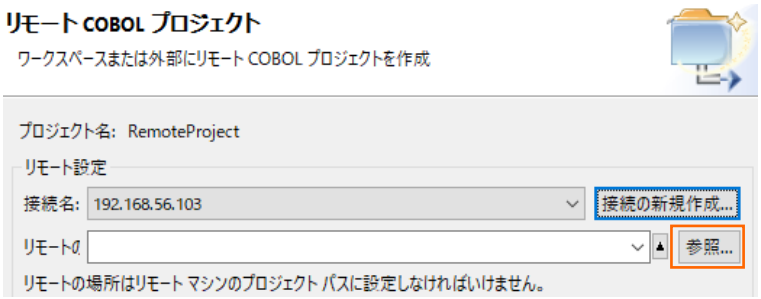
⑤ [Micro Focus DevHub(RSE 経由)] を選択し、[次へ(N)] ボタンをクリックします。



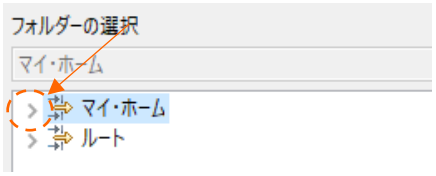
⑥ Windows 側にて Linux/UNIX サーバーの名前解決できるのであれば [ホスト名] 欄にそのホスト名を入力します。名前解決できない場合は、[ホスト名] 欄にはそのサーバーの IP アドレスを指定します。[接続名] 欄は自動で [ホスト名] 欄の値がコピーされます。指定が終わりましたら [終了(F)] ボタンをクリックします。



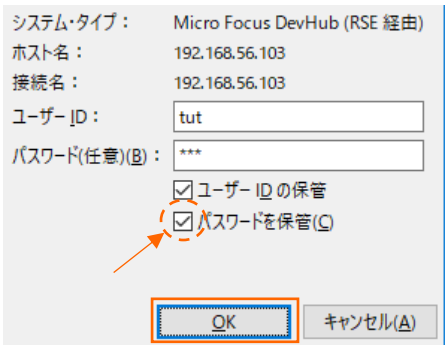
⑦ [参照...] ボタンをクリックします。



⑧ [マイ・ホーム] の左の展開アイコンをクリックします。

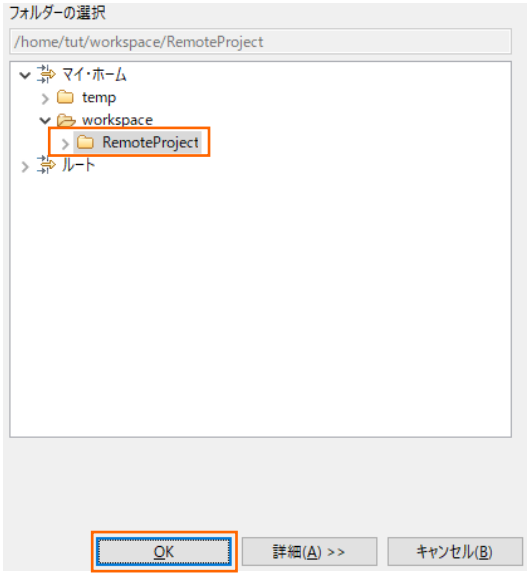


⑨ Linux/UNIX 側で利用する一般ユーザーの認証情報を [ユーザー ID] 欄及び [パスワード] 欄に入力します。[パスワードを保管(C)] にチェックを入れ、[OK] ボタンをクリックします。

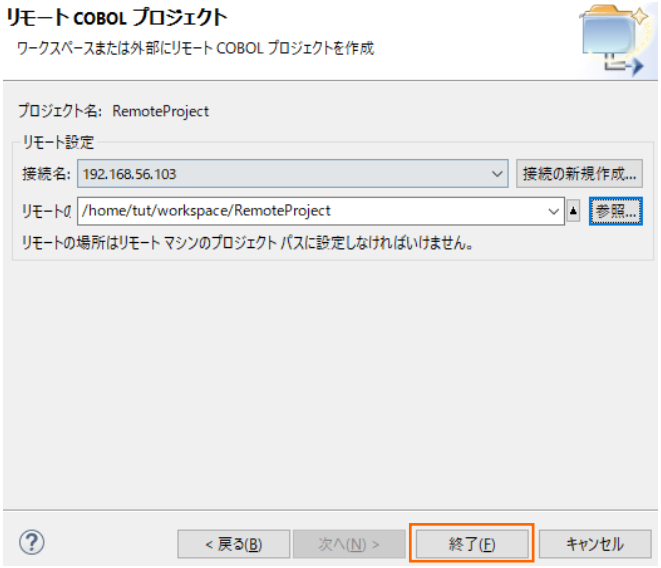


⑩ いくつかワーニングダイアログが表示されることがあります。その場合、すべての応答に [はい(Y)] をクリックします。

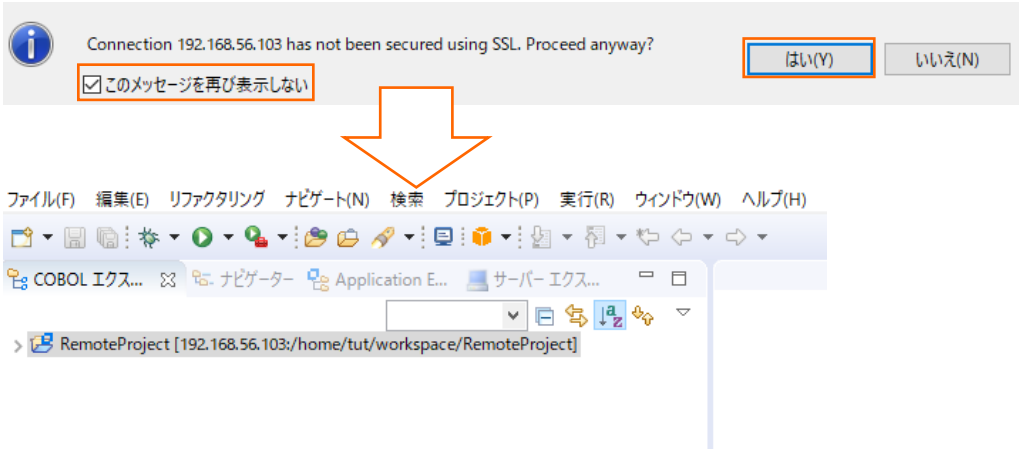
⑪ Linux/UNIX 側でソースや生成されるモジュール等を格納するプロジェクトディレクトリとして利用するディレクトリをツリーで選択し、[OK] ボタンをクリックします。



⑫ [終了(F)] ボタンをクリックします。



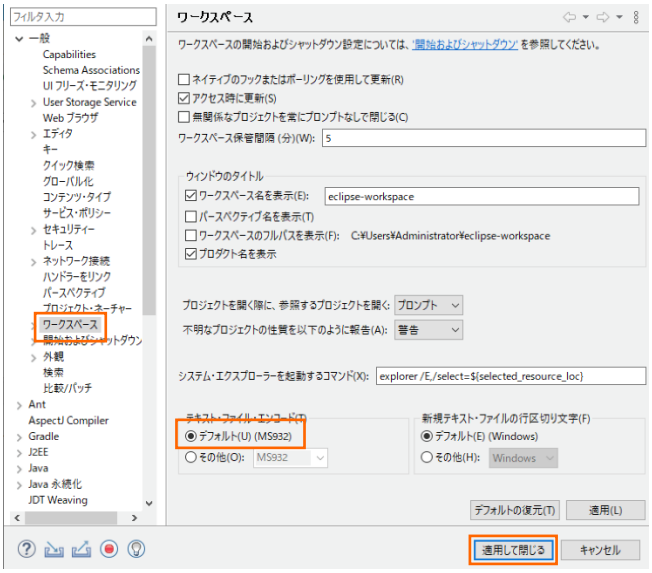
⑬　下記ダイアログが表示された場合、[このメッセージを再び表示しない] にチェックをいれ、[はい(Y)] を選択します。



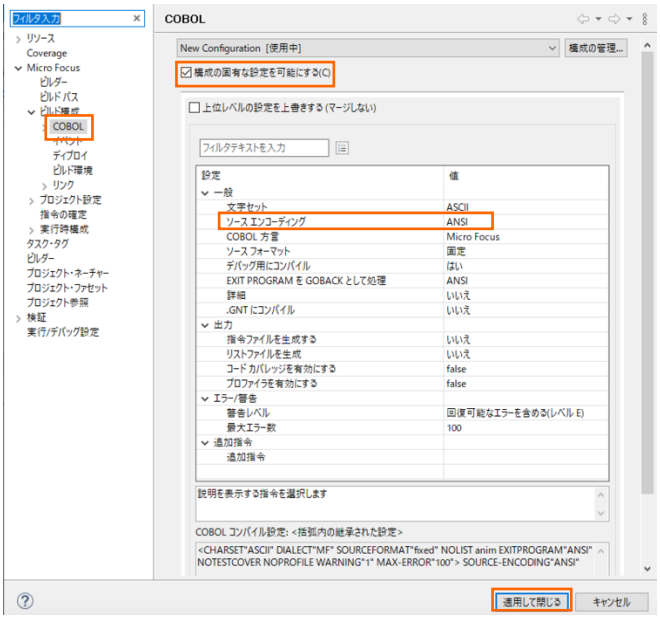
3)　文字コードの指定を行います。

Visual COBOL 9.0 より Shift-JIS を指定して日本語を表示する場合、文字コードの指定を明確に行う必要があります。

最初に、[Window(W)]メニュー ＞ [設定(P)]より[一般] ＞ [ワークスペース]とナビゲートし、テキストファイルエンコードを「MS932」に変更し、[適用して閉じる]ボタンをクリックします。

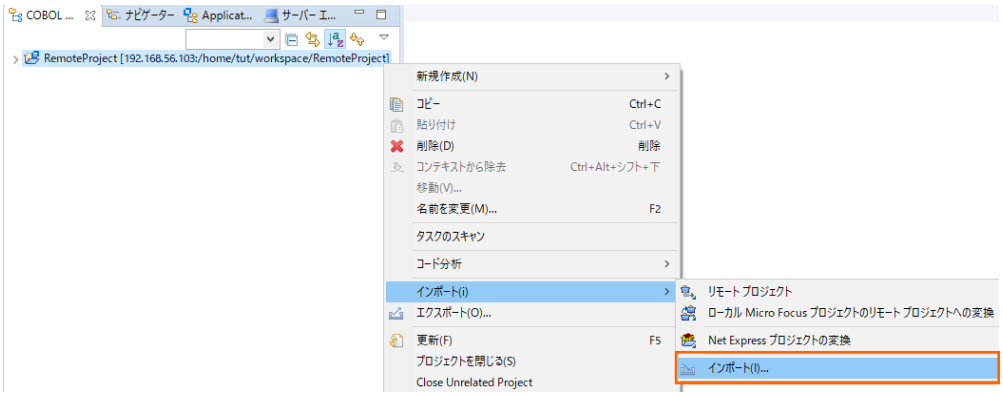


次に作成した COBOL プロジェクトを選択した状態で、マウスの右クリックにてコンテクストメニューを表示し、「プロパティ」を選択します。[Micro Focus] ＞ [ビルド構成] ＞ [COBOL]とナビゲートし、「構成の固有な設定を可能にする」にチェックを入れて[一般] ＞ [ソース エンコーディング]を”UTF-8”から”ANSI”に変更し、[適用して閉じる]ボタンをクリックします。

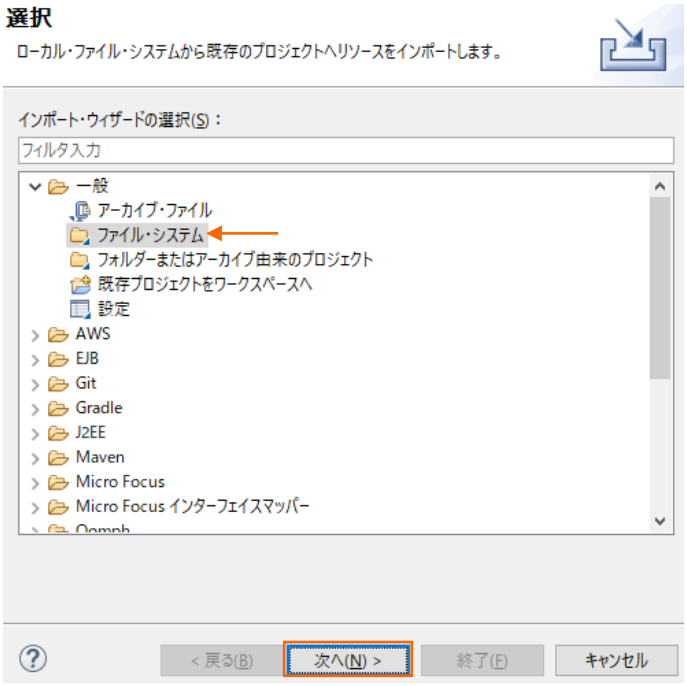


4) プロジェクトにリソースを追加します。

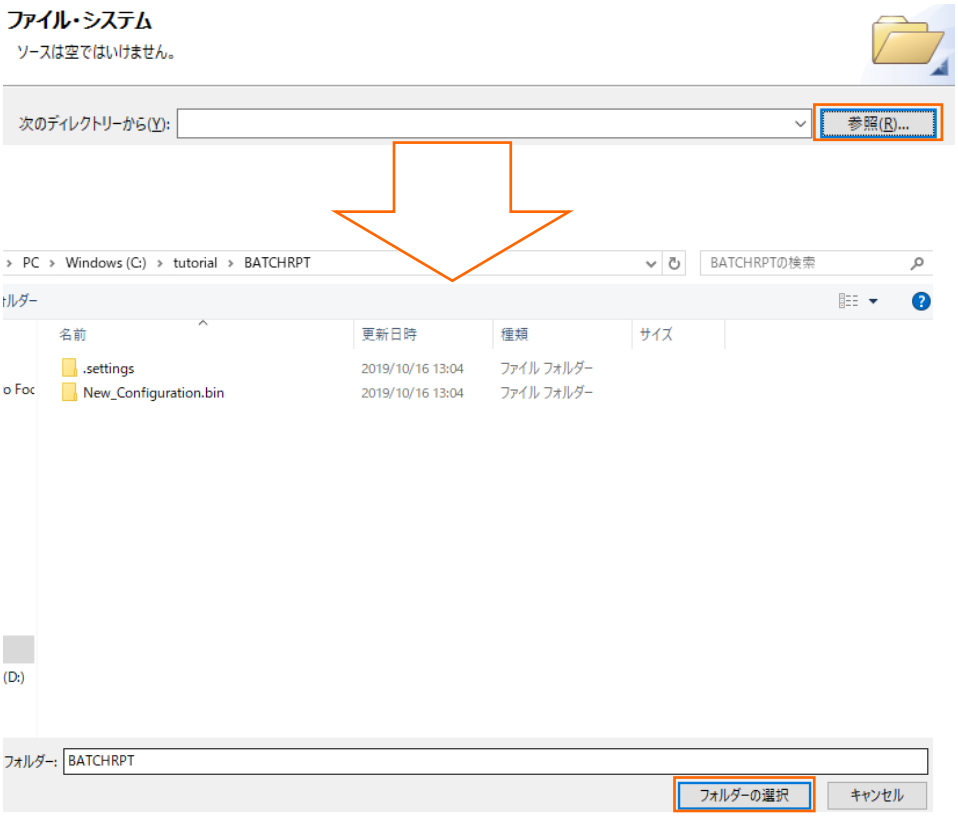
① プロジェクトを右クリックし、[インポート(I)] > [インポート(I)] を選択します。



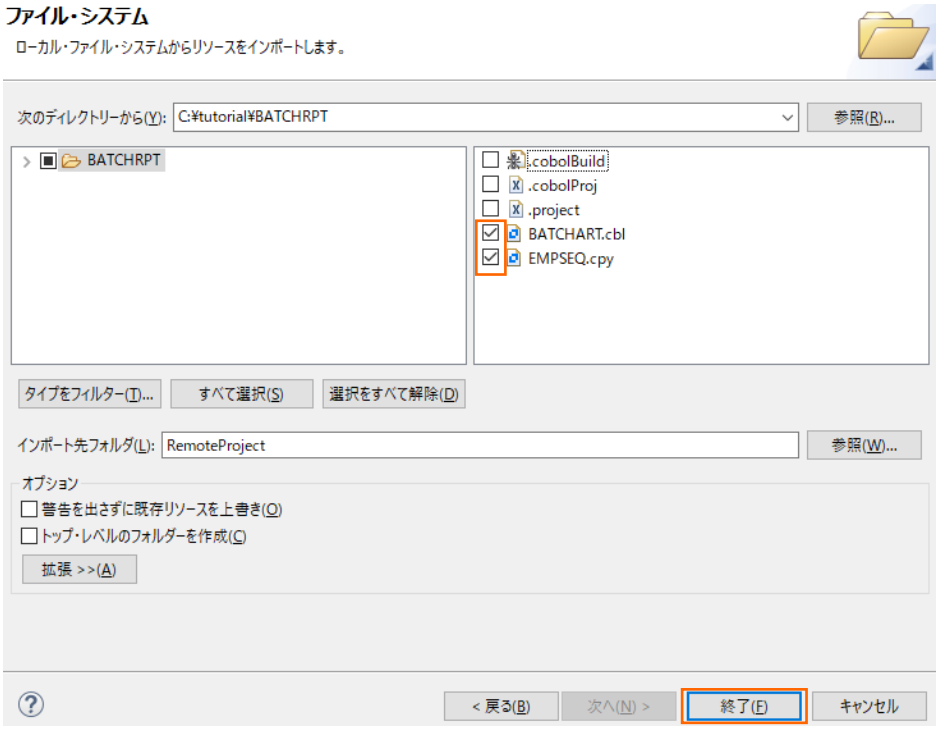
② [一般] > [ファイル・システム] を選択し [次へ(N)] ボタンをクリックします。



③ [参照(R)] ボタンをクリックし、ポップアップするエクスプローラにて「Visual COBOL for Eclipse 自習書」で作成した [BATCHRPT] プロジェクトフォルダを選択し [フォルダーの選択] ボタンをクリックします。

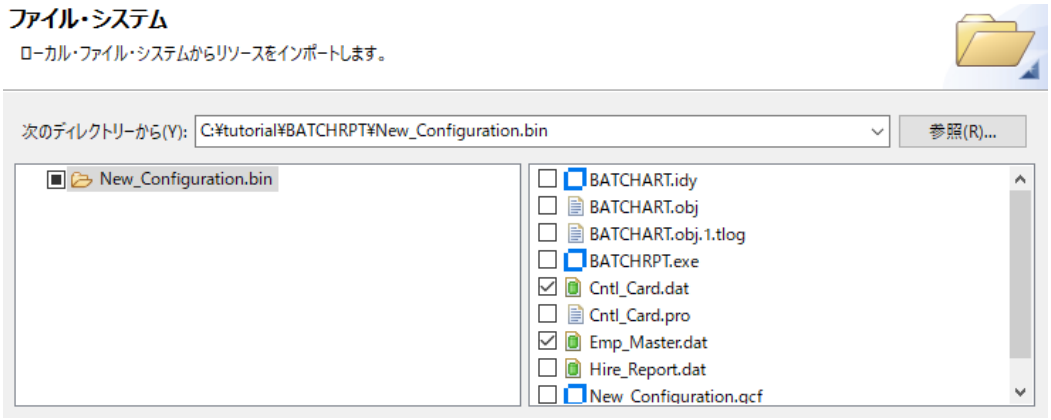


④ [BATCHRPT.cbl] 及び [EMPSEQ.cpy] にチェックを入れ、[終了(F)] ボタンをクリックします。

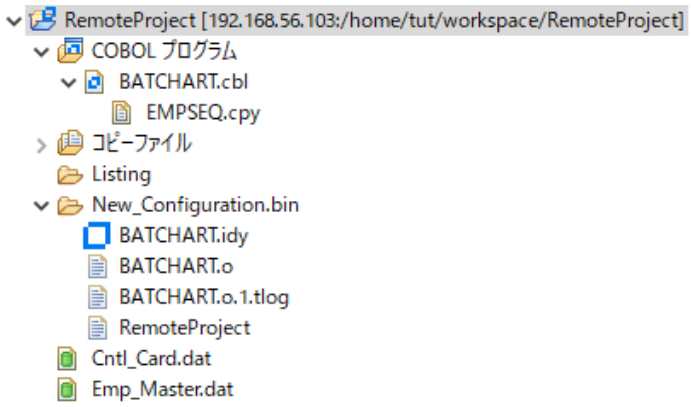

ファイル・システム
ローカル・ファイル・システムからリソースをインポートします。
次のディレクトリーから(Y): C:¥tutorial¥BATCHRPT
参照(R)...
BATCHRPT
.cobolBuild
.cobolProj
.project
BATCHART.cbl
EMPSEQ.cpy
タイプをフィルター(T)...
すべて選択(S)
選択をすべて解除(D)
インポート先フォルダ(L): RemoteProject
参照(W)...
オプション
警告を出さずに既存リソースを上書き(O)
トップ・レベルのフォルダーを作成(C)
拡張 >>(A)
< 戻る(B)
次へ(N) >
終了(F)
キャンセル

⑤ ③、④の要領で [Cntl_Card.dat] 及び [Emp_Master.dat] も BATCHRPT のプロジェクトフォルダ配下の New_Configuration.bin

⑥ フォルダ下から COBOL リモートプロジェクトに追加します



【リソース追加後のプロジェクトストラクチャーイメージ】

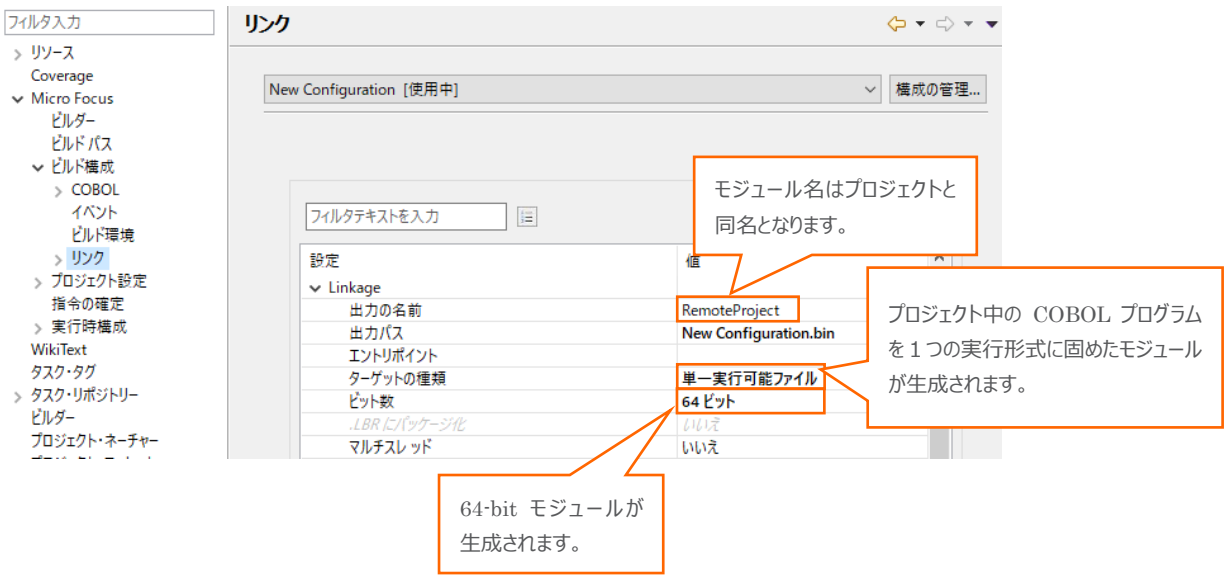


5) プロジェクト構成を設定します。

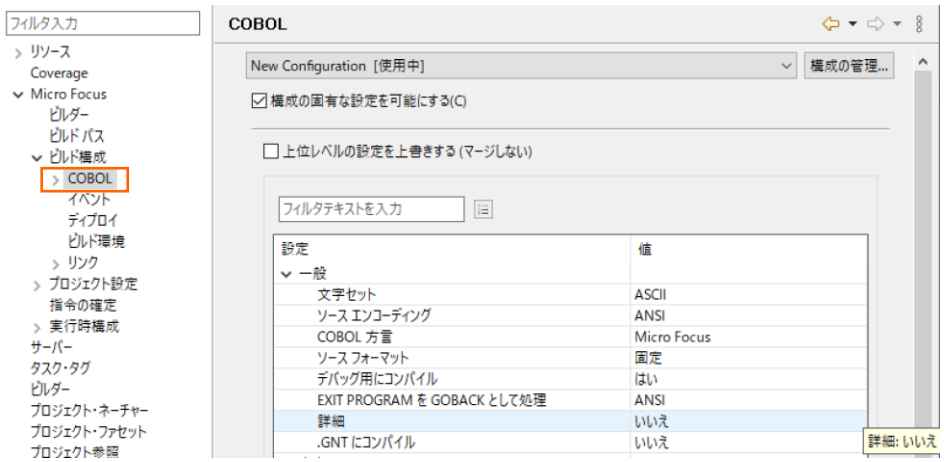
① COBOL エクスプローラにてプロジェクトを右クリックし、[プロパティ(R)] を選択します。

② [Micro Focus] > [ビルド構成] > [リンク] へとナビゲートします。

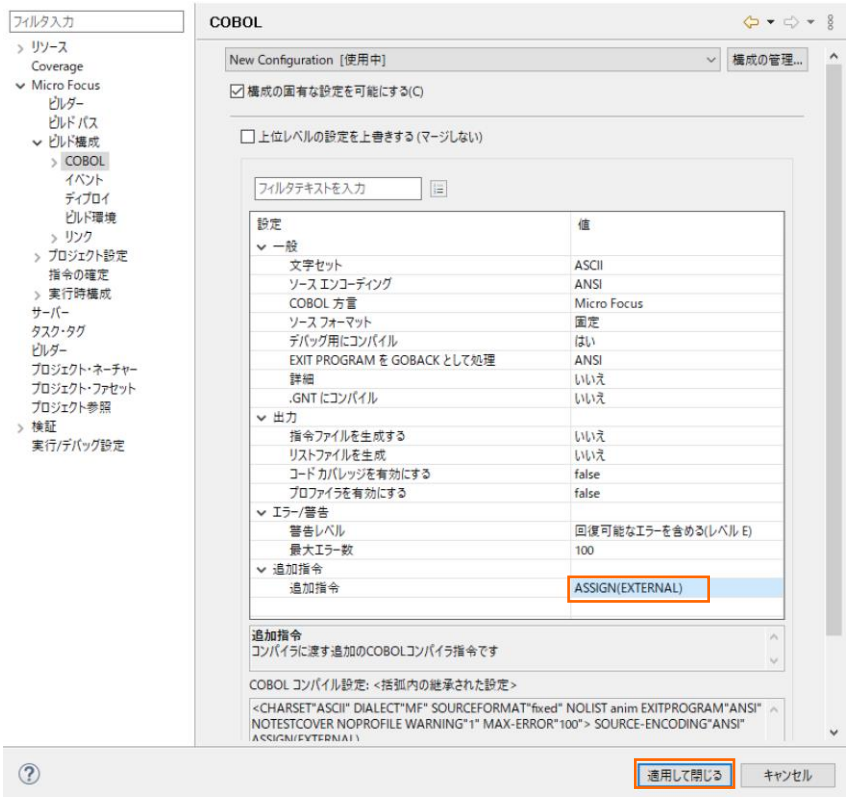
③ リンク設定を確認します。



④ [Micro Focus] > [プロジェクト設定] > [COBOL] へとナビゲートします。



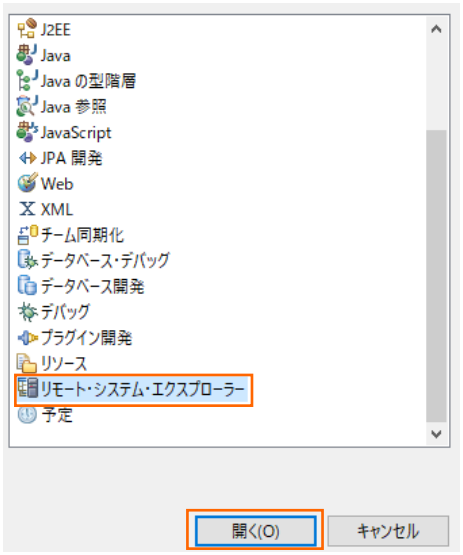
⑤ [追加指令] 欄に「ASSIGN(EXTERNAL)」を入力し [適用して閉じる] ボタンをクリックします。



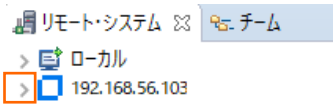
6) Linux/UNIX 上にリソースが生成されたことを確認します。

① 画面右上の [パースペクティブを開く] アイコンをクリックします。

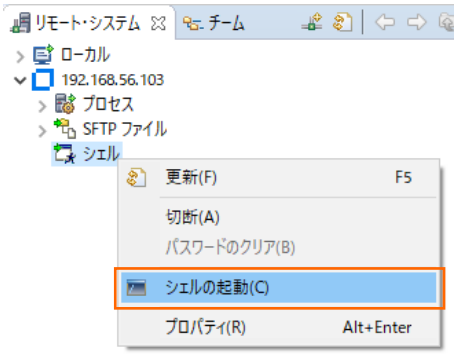
② [リモート・システム・エクスプローラー] を選択し、[開く(O)] ボタンをクリックします



③ [リモート・システム] ビューにて、プロジェクトを作成する際に作成した接続を展開します。



④ [シェル]を右クリックし、[シェルの起動(C)] を選択します。

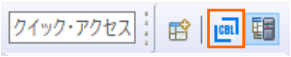

リモート・システム
チーム
ローカル
192.168.56.103
プロセス
SFTP ファイル
シェル
更新(F)
F5
切断(A)
パスワードのクリア(B)
シェルの起動(C)
プロパティ(R)
Alt+Enter

⑤ プロジェクトディレクトリとして用意したディレクトリの中身を確認します。

```
/home/tut/workspace/RemoteProject>
  ls -R
  .:
BATCHART.cbl
Cntl_Card.dat
EMPSEQ.cpy
Emp_Master.dat
Listing
New_Configuration.bin
  ./Listing:
  ./New_Configuration.bin:
  BATCHART.idy
  BATCHART.o
  BATCHART.o.1.tlog
RemoteProject
```

COBOL エクスプローラの表示が実際の Linux/UNIX 上のファイルシステム上の内容と同期がとれていることが確認できます。

⑥ 画面右上の COBOL パースペクティブのアイコンをクリックしてパースペクティブを COBOL に戻します。



3.5. X サーバーの準備

リモート開発でデバッグする際、ACCEPT 文や DISPLAY 文によるコンソール入出力は X Window の技術を用いて、Windows 側に表示させます。そのため、リモート開発にてデバッグ／テスト実行する際は、Windows 側で X サーバーを起動する必要があります。そのため Reflection Desktop for X という X サーバーも併せて配備する必要があります。Windows 端末上に既に他の X サーバーソフトをインストールしていればそれを利用することも可能です。未インストールの場合はこの Reflection Desktop for X をインストールしてリモート開発時に利用してください。

Reflection Desktop for X のインストーラはリリースノートに記載されておりますのでそちらを参照してください。

※「VcXsrv」と呼ばれる無償の X サーバーソフトも市場にあるのでそれを使うことも可能です。

3.6. リモートデバッグ

ここまでの作業にて、Windows 上の Eclipse プロジェクトから直接 Linux/UNIX 側に実行形式を生成させました。本章ではこの生成されたモジュールを Linux/UNIX 上で実行させつつも Windows 上のデバッガでその処理を操作してみます。

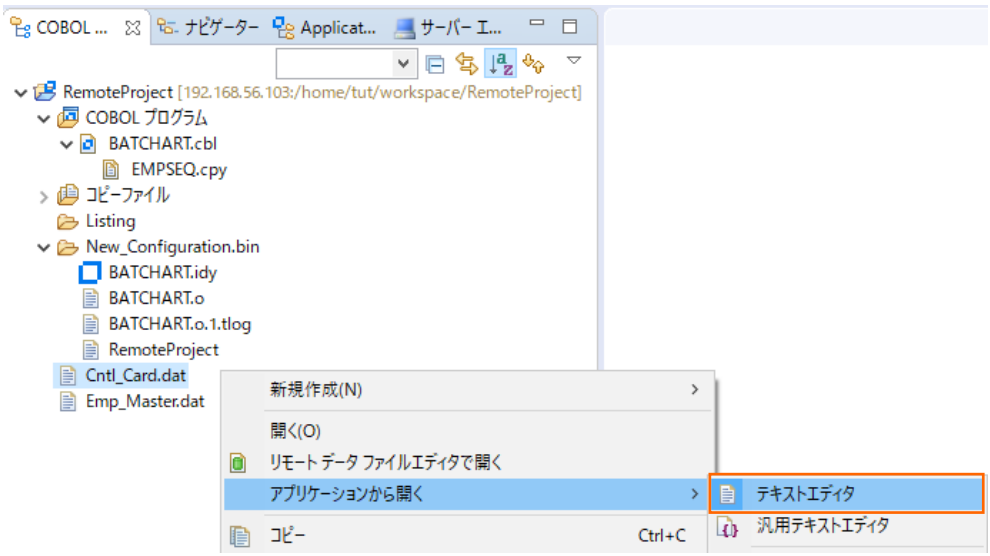
1) Visual COBOL for Eclipse が閉じている場合は、起動し 3.4 で使用した Eclipse ワークスペースを開きます。

2) ご使用になる X サーバーを起動します。

3) 制御ファイルのメンテナンスをします。

「Visual COBOL for Eclipse 自習書」では最終的に該当する社員情報が見つからなくなるようメンテナンスしました。ここでは初期値に戻し検索条件を有効にします。

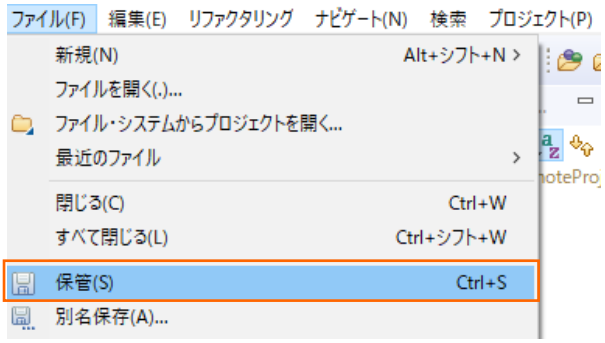
① COBOL エクスプローラにて [Cntl_Card.dat] を右クリックし、[アプリケーションから開く] > [テキストエディタ] を選択します。



② 「20110101」に変更します。

編集前　　編集後

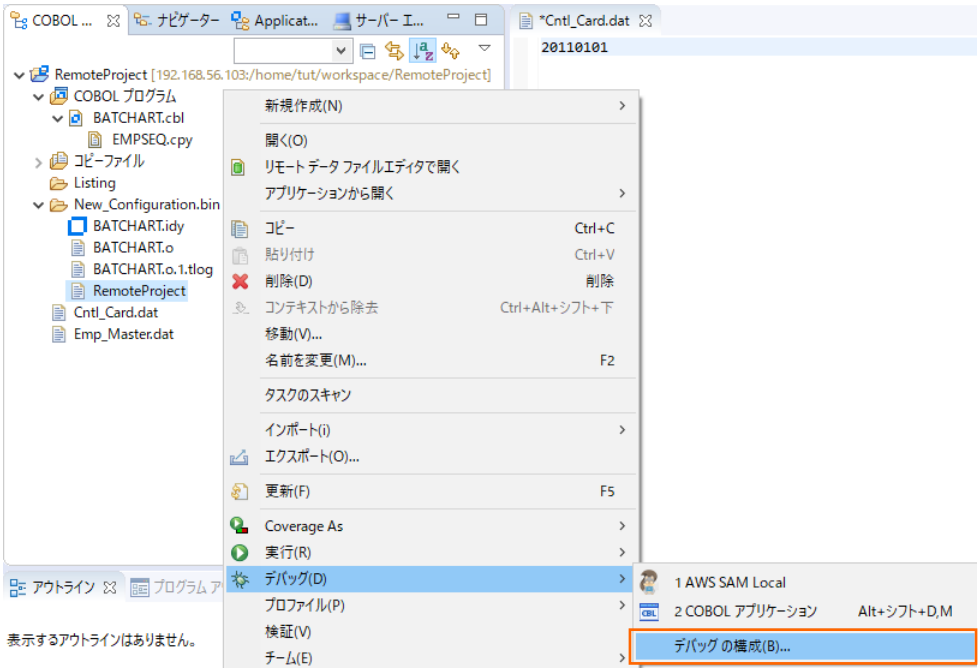
③ [ファイル(F)] メニューから [保管(S)] を選択し変更を保存します。



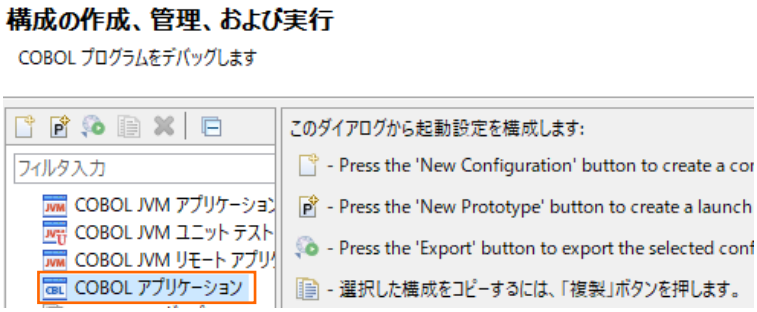
4) デバッグの構成の各種設定項目を指定します。

① COBOL エクスプローラにて [New_Configuration] 配下に生成されているプロジェクトと同名の実行形式を右クリ

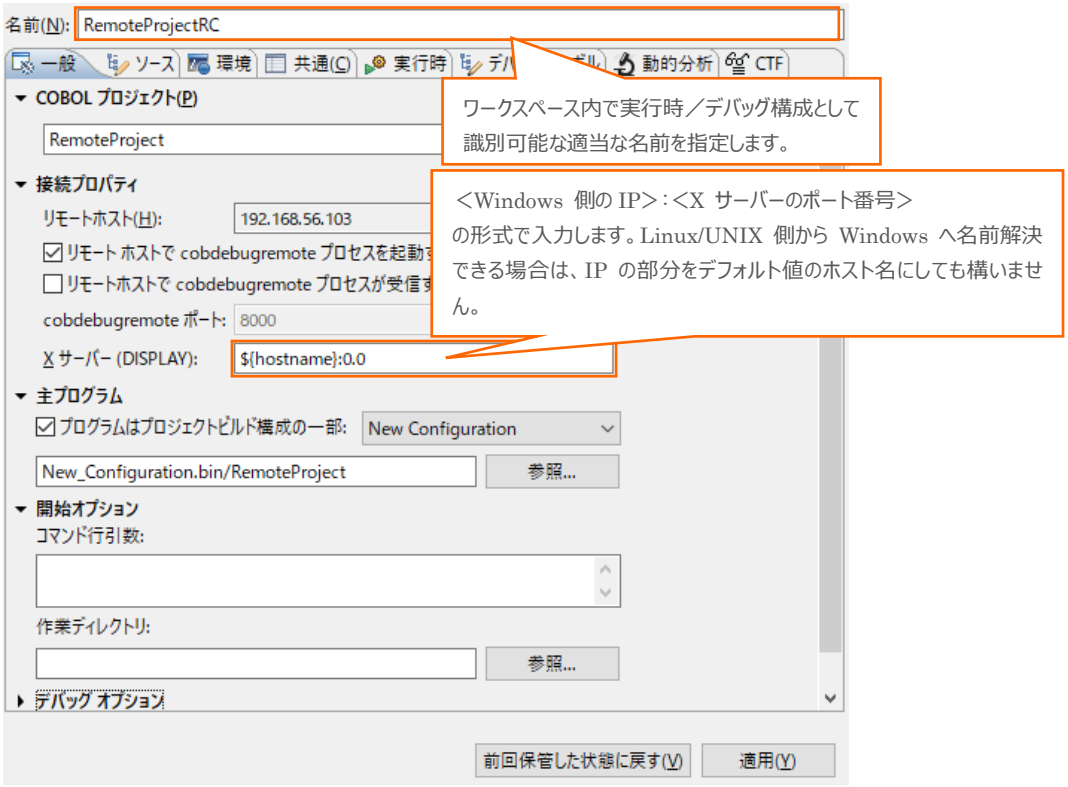
ックし [デバッグ(D)] > [デバッグの構成(B)] を選択します。



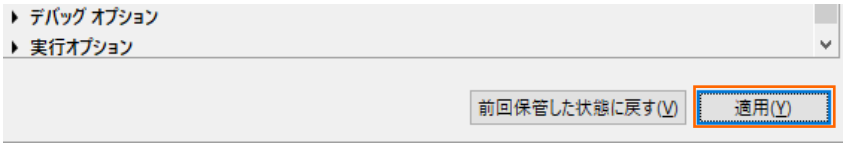
② [COBOL アプリケーション] をダブルクリックします。



③ [名前] 欄及び [X サーバー(DISPLAY)] 欄へ値を設定します。

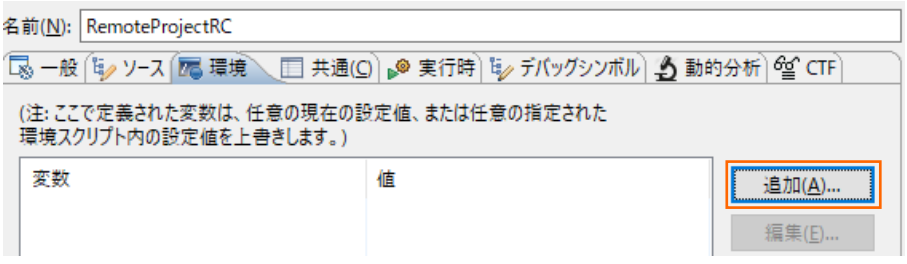


④ [適用(Y)] ボタンをクリックし変更を保存します。



⑤ [環境] タブをクリックします。

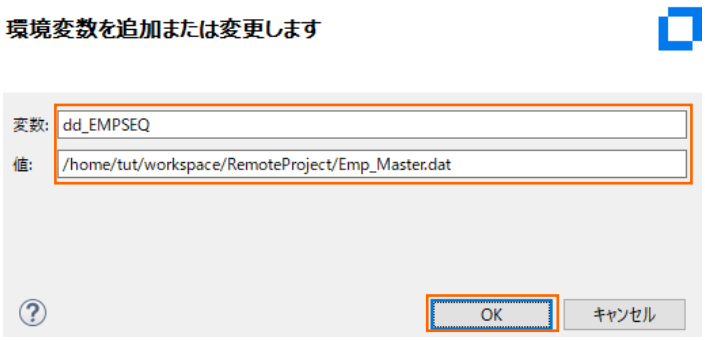
⑥ [追加(A)] ボタンをクリックします。



⑦ 下記のように入力し [OK] ボタンをクリックします。

変数： dd_EMPSEQ

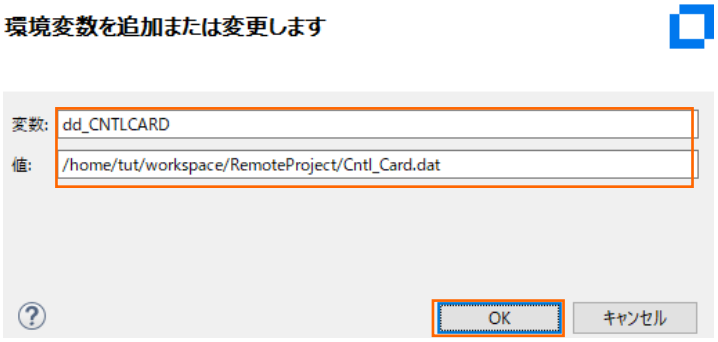
値： Emp_Master.dat までのフルパス



⑧ ⑥、⑦の要領で下記のエントリも追加します。

変数： dd_CNTLCARD

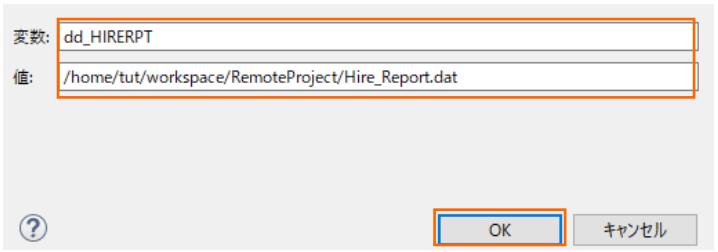
値： Cntl_Card.dat までのフルパス



⑨ 更に同様に下記のエントリも追加します。

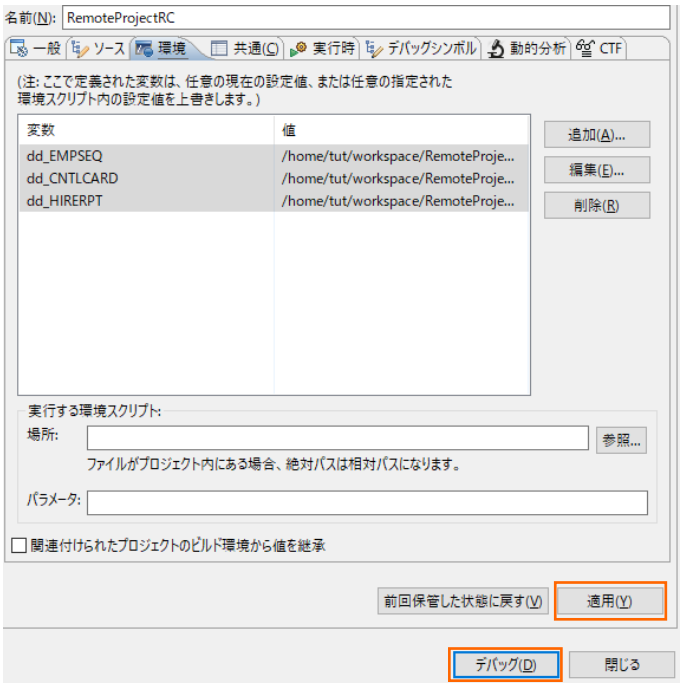
変数： dd_HIRERPT

値： <プロジェクトディレクトリ> /Hire_Report.dat



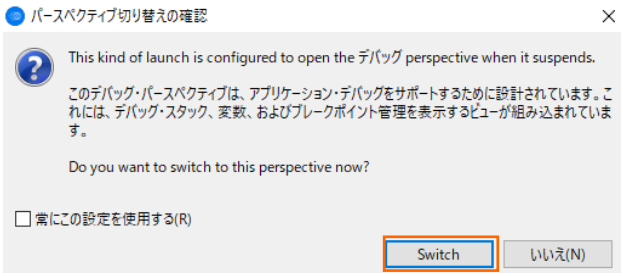
5) デバッグ実行を開始します。

前のステップで指定したデバッグ構成ウィンドウにて［適用(Y)］ボタンに続き［デバッグ(D)］をクリックしデバッグ実行を開始します。

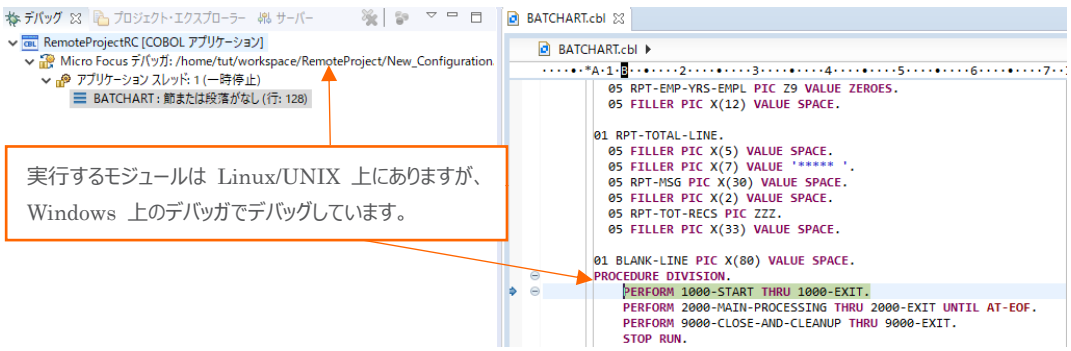


6) Eclipse 上のデバッガを使ってデバッグします。

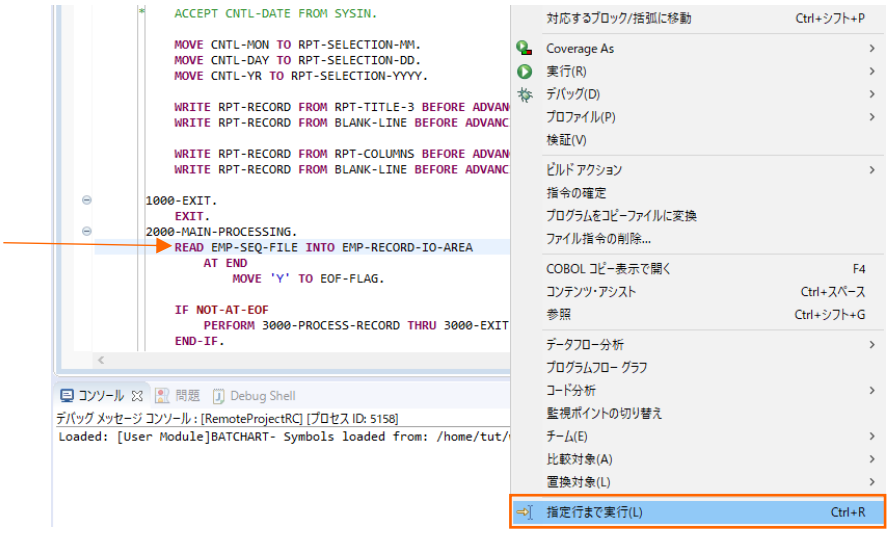
① パースペクティブの切り替えに関するメッセージに関しては［Switch］ボタンをクリックしてデバッグパースペクティブへ切り替えます。



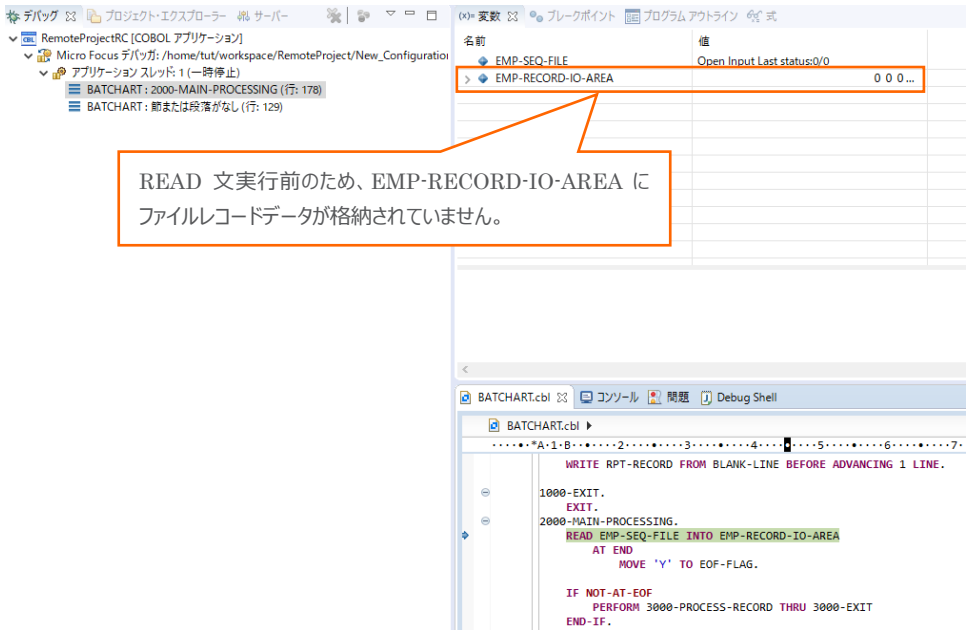
最初の COBOL 行の実行前で処理が一時停止しています。



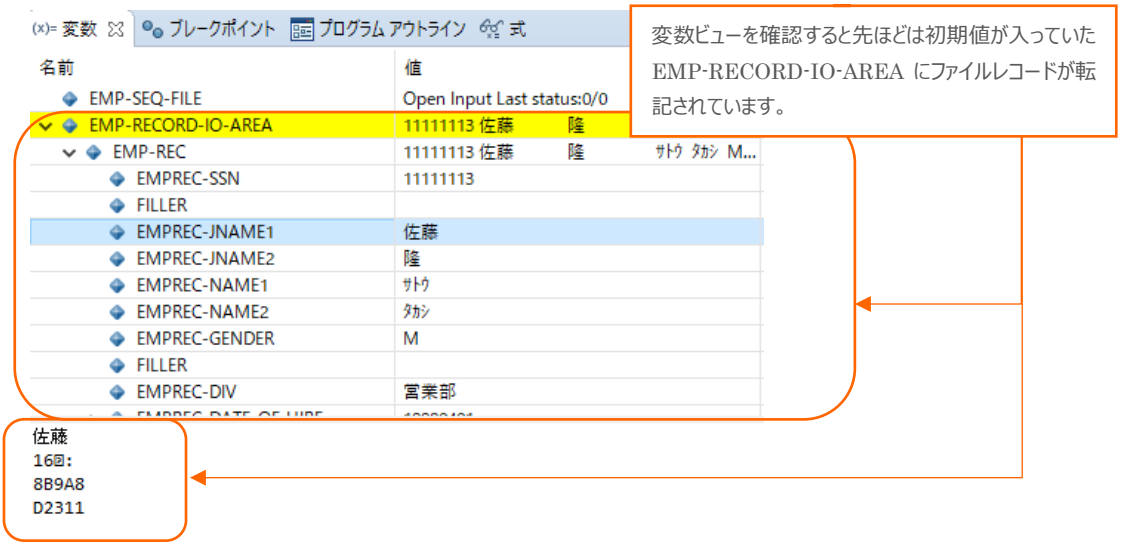
② ［2000-MAIN-PROCESSING］段落の最初の READ 文にカーソルを合わせ、右クリックから［指定行まで実行(L)］を選択します。



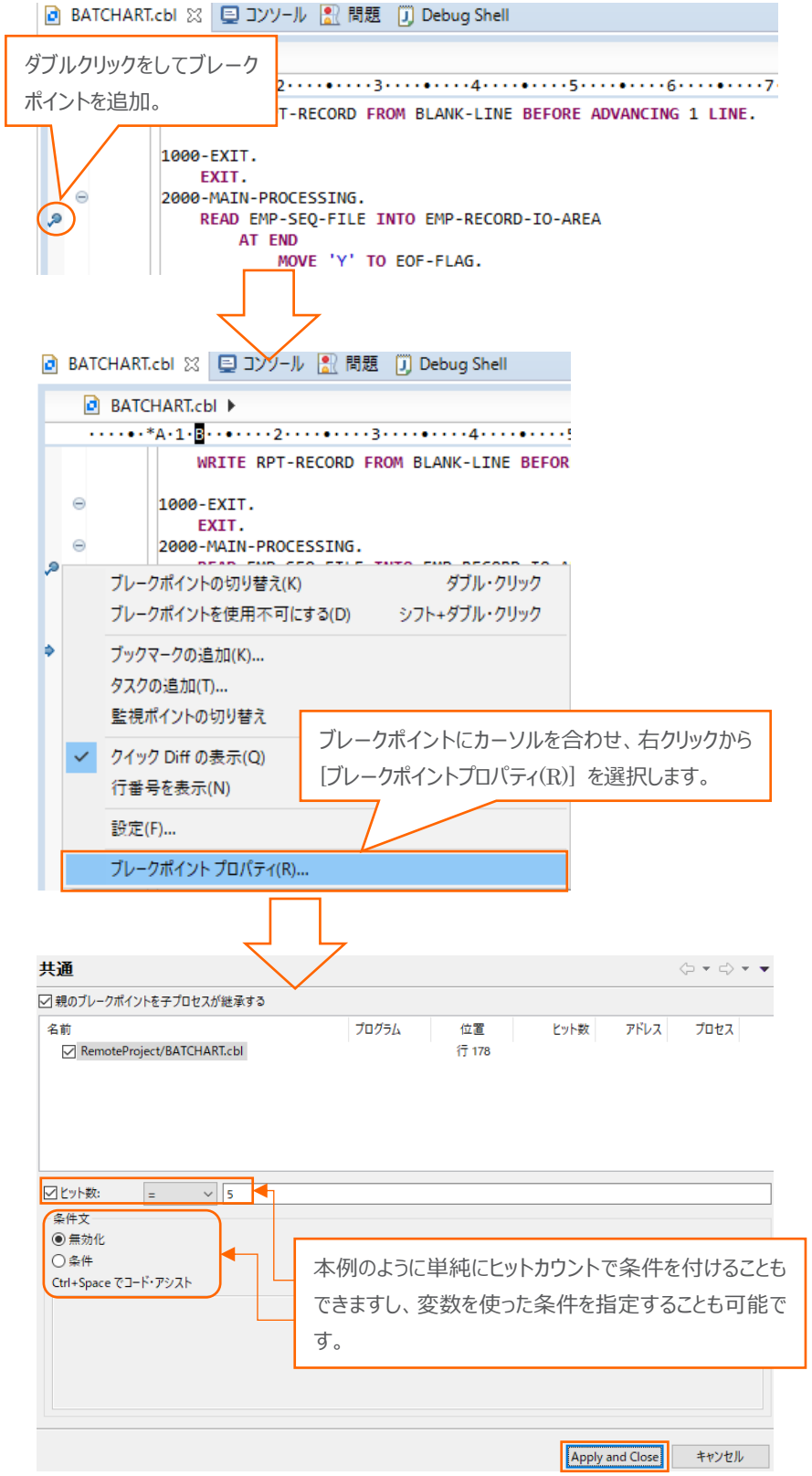
カーソル位置まで処理が進みます。



③ F5 を打鍵し、READ 文を実行します。

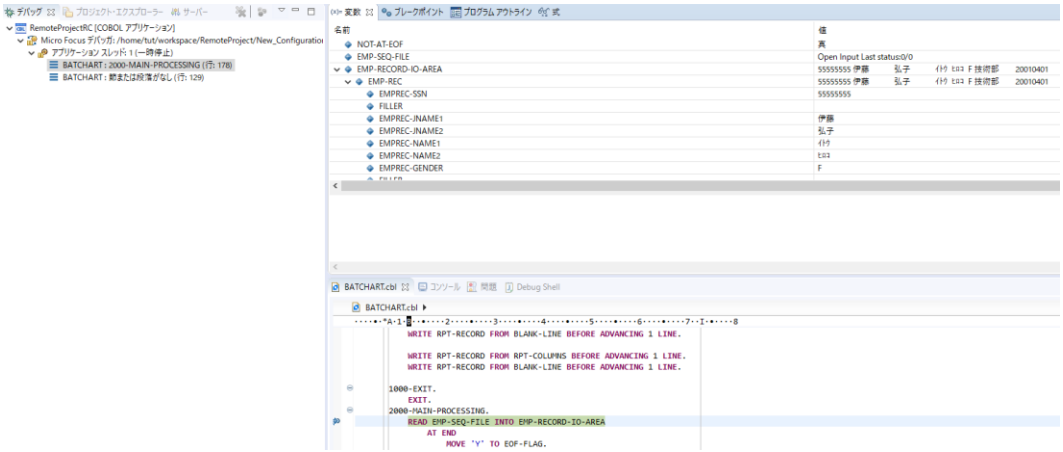


④　条件付きブレークポイント機能を確認します。



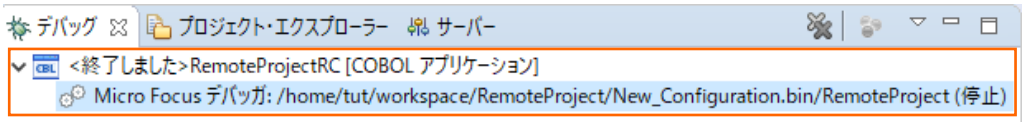
[Apply and Close] ボタンをクリックして、ヒットカウントによる条件付きブレークポイントを設定します。

設定後、F8 を打鍵しますと、設定した直後から５回目の READ 文のヒットでデバッガが一時停止します。



⑤ デバッガの動作が確認できましたら、F8 を打鍵しアプリケーションを最後まで実行します。

デバッガが終了した旨を [デバッグ] ビューより確認できます。

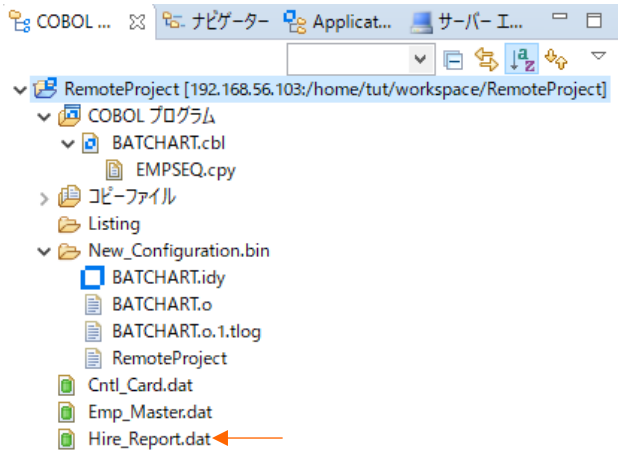


7) COBOL パースペクティブに戻します。

画面右上の COBOL パースペクティブアイコンを選択します。

8) 生成された帳票を確認します。

① COBOL エクスプローラにて Hire_Report.dat が生成されていることを確認します。確認できない場合はプロジェクトを右クリックの上、[更新(F)] を選択し、ビューをリフレッシュします。



② COBOL エクスプローラ中の Hire_Report.dat を右クリックし、[アプリケーションから開く] > [テキストエディタ] を選択します。

「Visual COBOL for Eclipse 自習書」で確認したのと同じ帳票が生成されていることが確認できます[6]。

以上で本チュートリアルは完了です。

チュートリルを終了する際、Eclipse はそのまま閉じていただいて構いません。

Linux/UNIX 側は Windows 側の Eclipse が終了した後に

$COBDIR/remotedev/stoprdodaemon

を実行しデーモンを終了させます。

WHAT'S NEXT

- 本チュートリアルで学習した技術の詳細については製品マニュアルをご参照ください。

免責事項

[6] Eclipse におけるデフォルトのテキストエディタフォントがプロポーショナルになっている場合は多少見た目が異なる可能性があります。この場合、テキストエディタ上で右クリックから［設定］を選択し［一般］> [外観] > [色とフォント］で表示されるページにてフォントを変更できます。