

Dcs_(Ver. 2.10)

ビルド・ガイド

目次

概要	3
1. DCS (DLL) プロジェクト	
1-1. DCSプロジェクト生成	5
1-2. DCSプロジェクトにソース・ファイル登録	7
1-3. DCSプロジェクトのプロパティー設定	8
1-4. DCSプロジェクトのビルド	11
2. DCSメイン (EXE) プロジェクト	
2-1. DCSメイン・プロジェクト生成	12
2-2. DCSメイン・プロジェクトにソース・ファイル登録	13
2-3. DCSメイン・プロジェクトのプロパティー設定	13
2-4. DCSメイン・プロジェクトのビルド	14
2-5. DCSメインの実行	14
3. JDCS (DLL) プロジェクト	
3-1. JDCSプロジェクト生成	19
3-2. JDCSプロジェクトのプロパティー設定	19
3-3. JDCSプロジェクトのビルド	21
4. JDCS (JAR) プロジェクト	
4-1. JDCSプロジェクト生成	22
4-2. JDCSプロジェクト・ビルド	24
4-3. JDCSプロジェクト JAR 生成	25
5. DCSビューアー／JDCSテスト・プロジェクト	
5-1. DCSビューアー／JDCSテスト・プロジェクト・ビルド	27
5-2. DCSビューアー用 JARファイルの登録	28
5-3. DCSビューアー／JDCSテストの実行	30
5-4. DCSビューアー実行用 JAR 生成／実行	31
6. Linux	
6-1. DCSプロジェクト	32
6-2. DCSビューアー・プロジェクト	35
7. Android—Termux, UserLAnd (Ubuntu)	
7-1. リソース dcs_build_src のSDカードへのコピー	36
7-2. Androidでのビルド	37
7-3. Termux でのビルド	38
7-4. UserLAnd (Ubuntu) でのビルド	39
8. Cygwin	
8-1. インクルード・パスの設定	41
8-2. DCSのビルド	41

概要

dc_build_src.zipからD c sをビルドする方法をプラットフォームごとに解説します。Mac は Linux と同様にビルドできるかも知れませんが対象外です。ここに示す生成方法はあくまでも一例です。環境に合った適切な方法を考慮して下さい。生成した共有ライブラリー、コマンド等の使用方法は「Dcs 使用者 ガイド」(dcs_users_guide_v210.pdf)を参照して下さい。

Windows, Linuxでのビルド方法

Eclipse によりプロジェクトを新規に生成して行なう

Cygwin, Android(Termux, UserLAnd)でのビルド方法

dc_build_src.zip に含まれる GCC, JDK 等を用いるシェル・スクリプトにより行なう

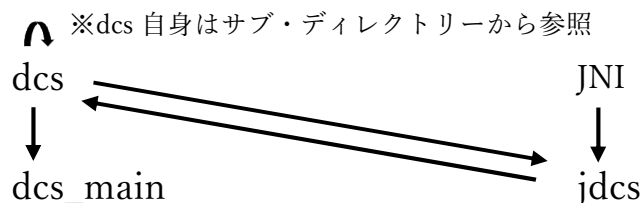
dc_build_src.zip: ビルド用の全ソース・ファイルとシェル・スクリプト

ディレクトリー構成 (青字がソース・ファイル)

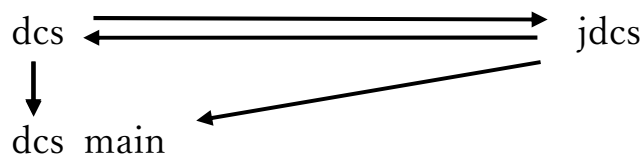
```
dc_build_src/ ルート
  cpp/      dcs/ DCS ライブラリー (C言語)
            dcs_main/ DCS メイン (C言語)
            jdcs/ DCS JNI (C言語)
  java/     jdcs/ JNI アクセス用 jar (Java)
            jdcs_test/ JDCS テスト・プログラム (Java)
            jdcsview/ 文字コード判定／変換 (Java)
  cygwin/   シグウィン用ビルド・シェル
  android/ アンドロイド(Termux, UserLAnd)用ビルド・シェル
```

依存関係

インクルード・ファイル



ライブラリー



想定ビルド・パス

Windows:

dcx_build_src: C:\Users\rabbit\Documents\dcx_build_src
Workspace C: C:\Users\rabbit\ws2109cpp_test
Workspace Java: C:\Users\rabbit\ws1912_test

Cygwin:

dcx_build_src: /cygdrive/c/Users/rabbit/Documents/dcx_build_src

Linux:

dcx_build_src: /home/rabbit/src/dcx_build_src
Workspace C: /home/rabbit/ws2109cpp_test
Workspace Java: /home/rabbit/ws2109_test

Android Termux:

Windows 側SDカード: G:\Android\data\com.termux\files\dcx_build_src
Android 側SDカード: ~/storage/external-1/dcx_build_src

Android UserLAnd(Ubuntu):

Windows 側SDカード: G:\Android\data\tech.ula\files\storage\dcx_build_src
Android 側SDカード: /storage/sdcard/dcx_build_src

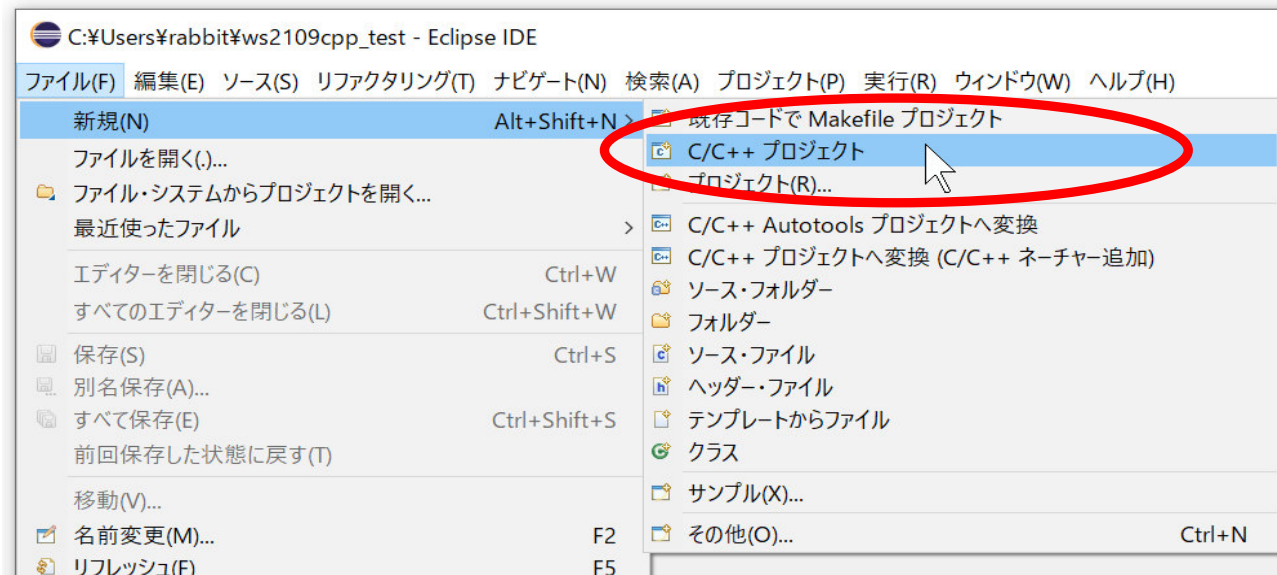
※ソース・ファイルは自由にカスタマイズして構いませんが、以下の形式である先頭のコメントは残して下さい。

```
/*  
 * <ソース・ファイル名>  
 *  
 * Copyright 2019-2022 Ichiji Tadokoro. All Rights Reserved.  
 */
```

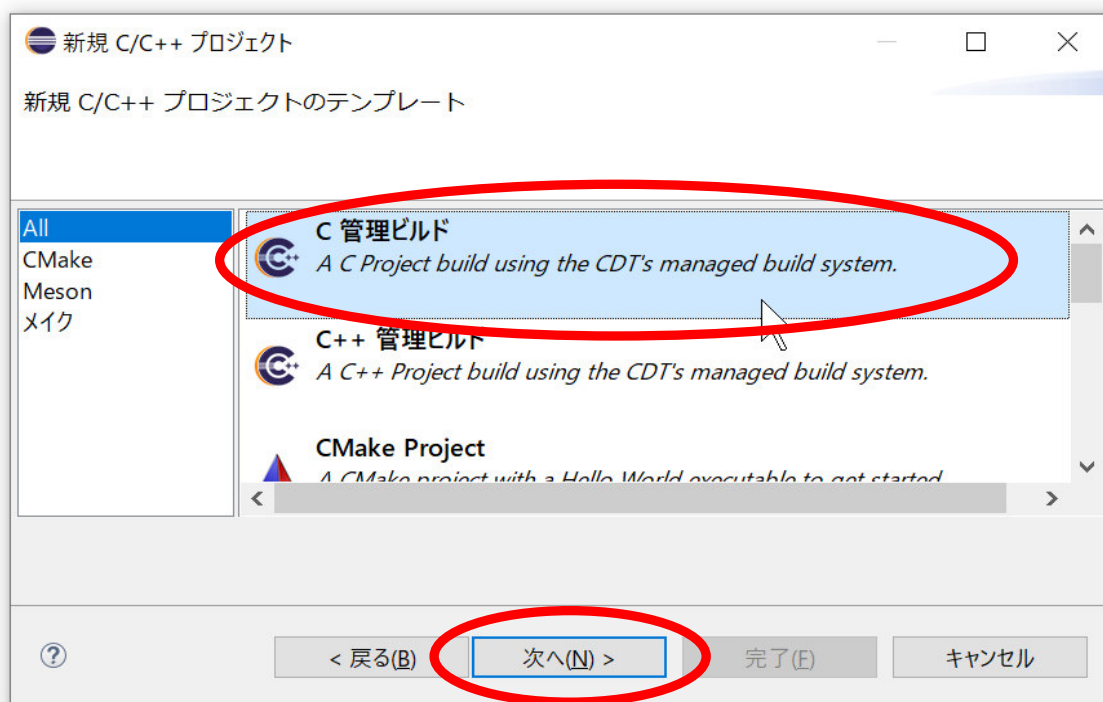

1. DCS(DLL)プロジェクト

1-1. DCSプロジェクト生成

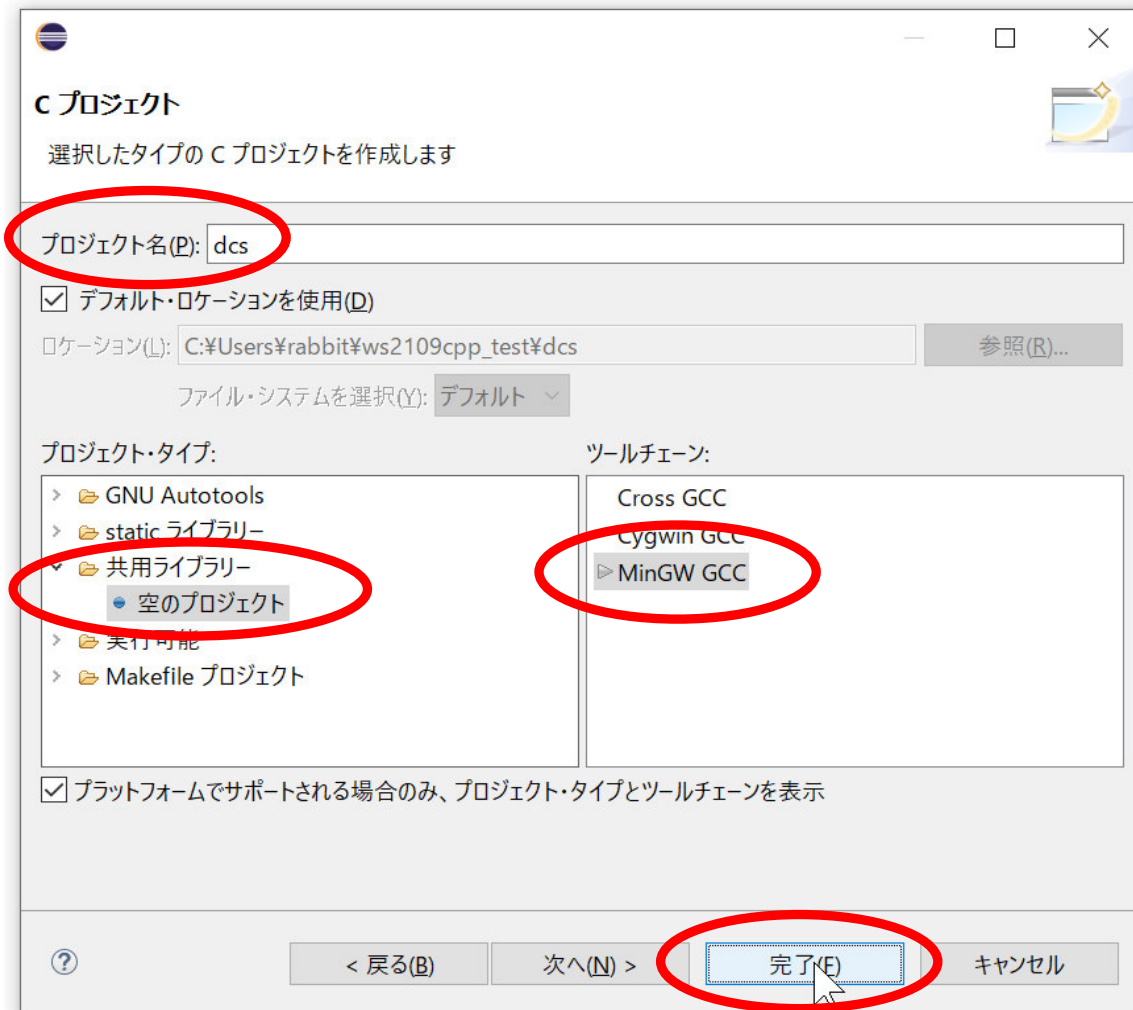
メニュー・バーより「ファイル」「新規」「C/C++ プロジェクト」を選択



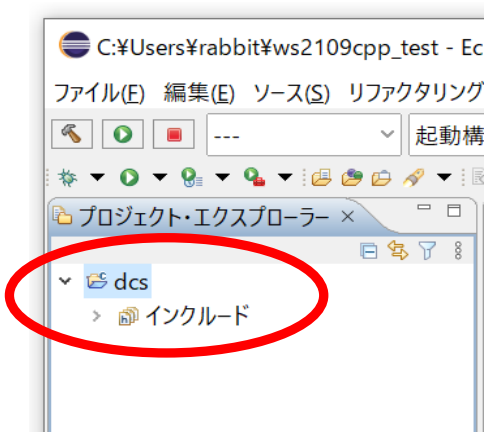
ダイアログより「C 管理ビルド」を選択して「次へ」



ダイアログより「プロジェクト名」を入力、「プロジェクト・タイプ」で「共用ライブラリー」「空のプロジェクト」を選択、「ツールチェーン」で「MinGW GCC」を選択し「完了」—Linux 系では先頭の「Cross GCC」がデフォルトなので注意！



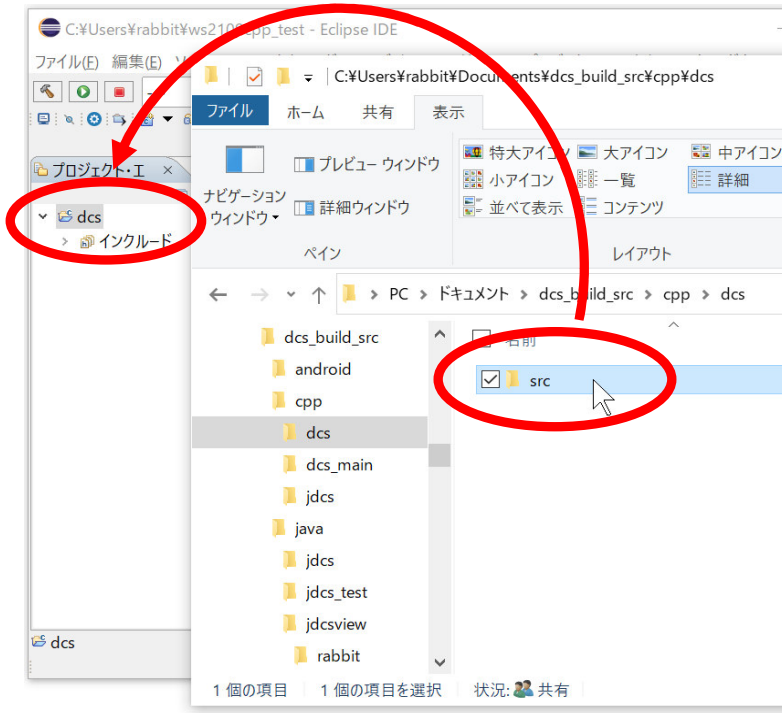
プロジェクト dcs が生成される



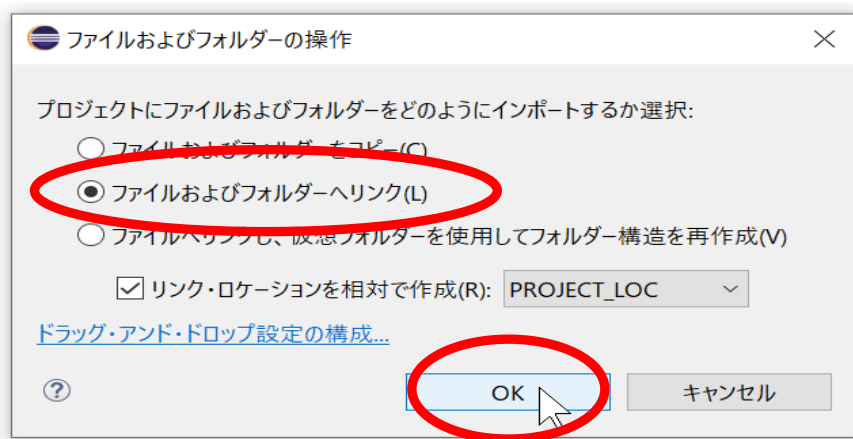
1 - 2. DCSプロジェクトにソース・ファイル登録

ソース・ファイルの登録

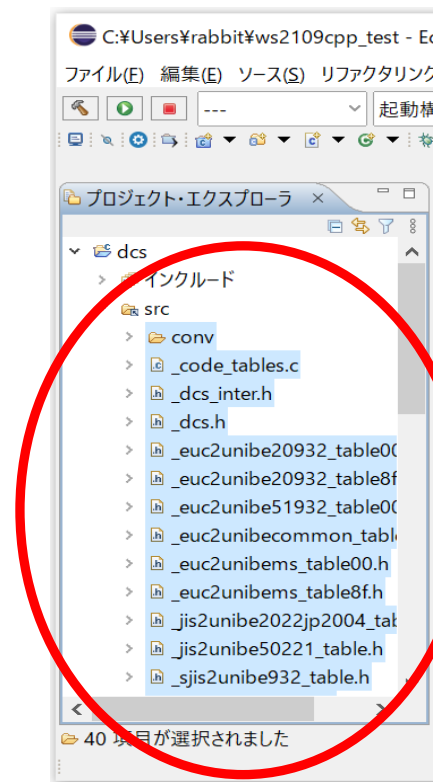
DCS ライブラリー配下の src を選択し、生成した dcs プロジェクト・フォルダーへドラッグ&ドロップ



ダイアログより「ファイルおよびフォルダーへリンク」を選択（他の選択肢も可）して「OK」

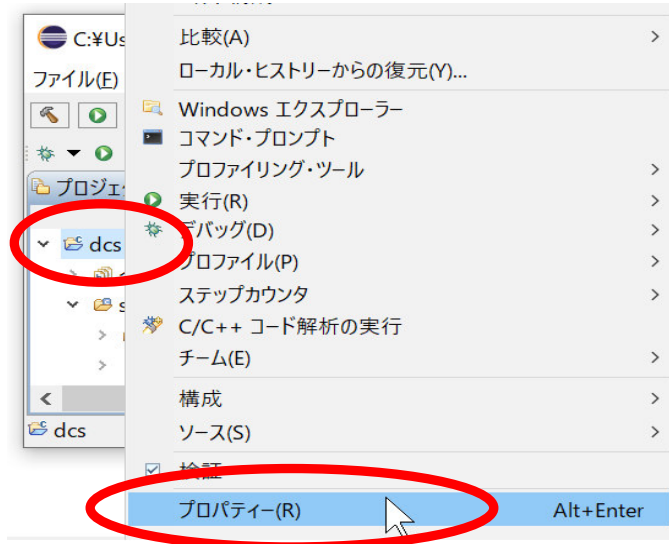


ソース・ファイルが登録されたことを確認



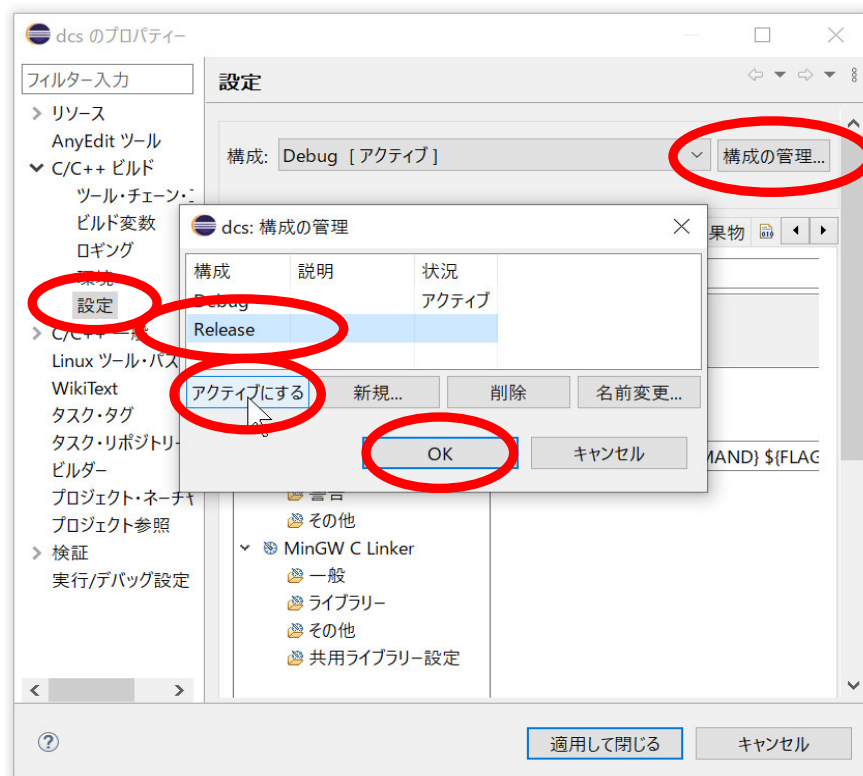
1 - 3. DCSプロジェクトのプロパティ設定

プロジェクト名 dcs を選択し、マウス右クリックでポップアップメニューより「プロパティ」を選択



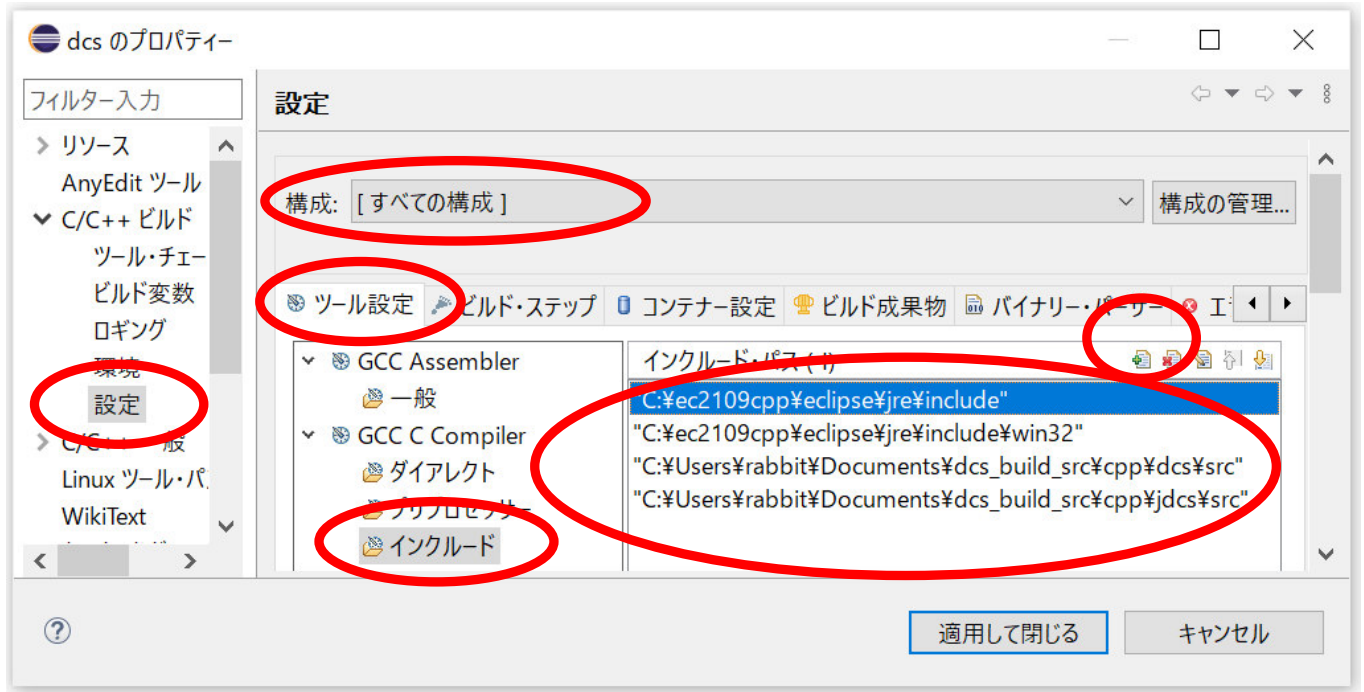
構成 Release のアクティブ化

ダイアログより「C/C++ ビルド」「設定」を選択し「構成の管理…」をクリックして「構成の管理」ダイアログより **Release** を選択し「アクティブにする」をクリックして「OK」



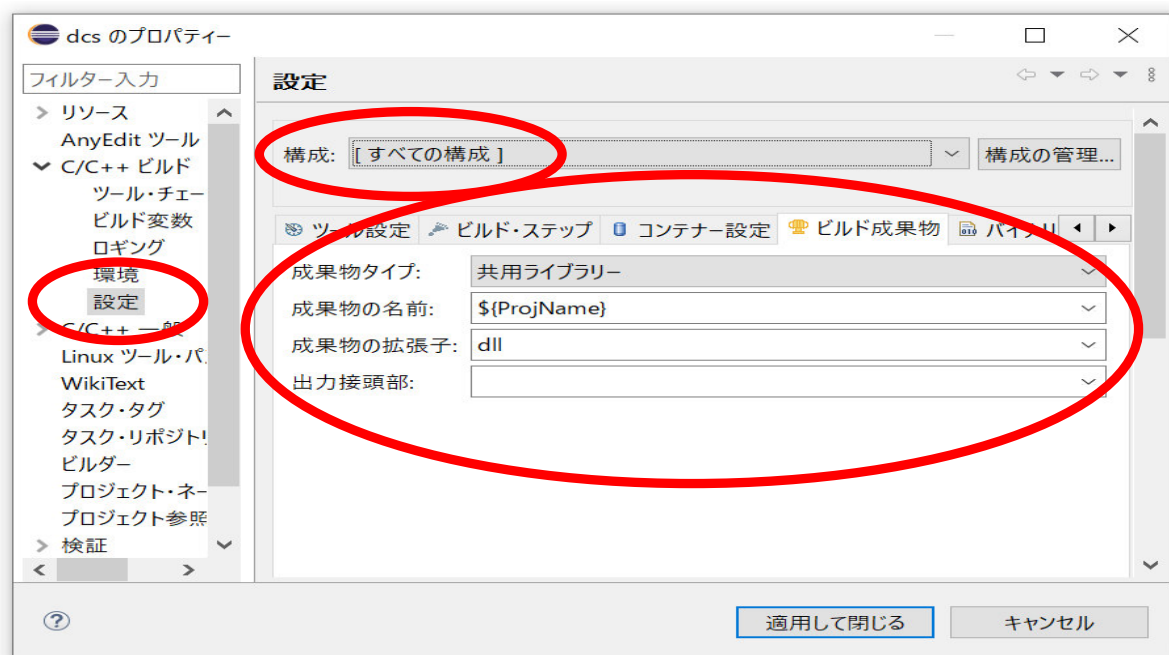
インクルード・パスの設定

ダイアログより「C/C++ ビルド」「設定」を選択し「構成」「すべての構成」を選択し「ツール設定」タブより「GCC C Compiler」「インクルード」を選択し「インクルード・パス」に「追加…」アイコンをクリックして dcs, jdc の srcパスを登録。さらに JRE配下にあるJNI用の2パスを設定。JREのバージョンは 1 1 以上であれば可。ここでは Eclipse のものを使用。



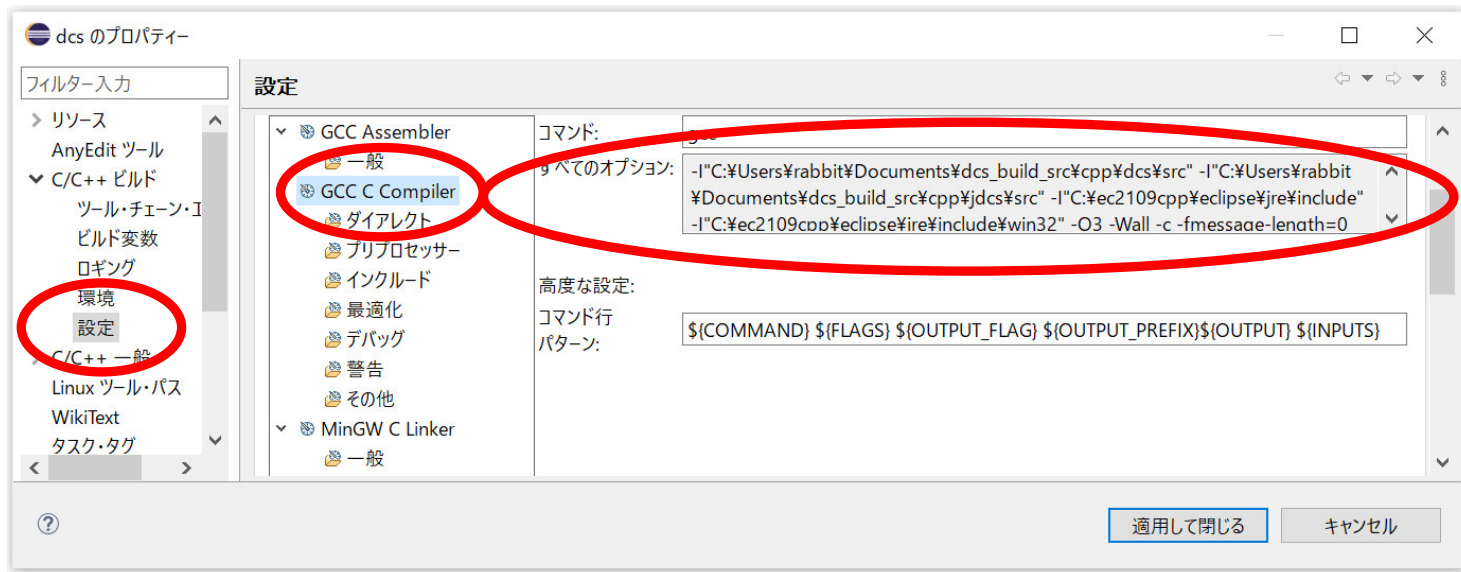
DLL名の設定

ダイアログより「C/C++ ビルド」「設定」を選択し「構成」「すべての構成」を選択し「ビルド成果物」タブより「成果物タイプ」「共用ライブラリー」、「成果物の名前」dcs(\${ProjName})、「成果物の拡張子」dll、「出力接頭部」を削除



コンパイラー・オプションの確認

ダイアログより「C/C++ ビルド」「設定」を選択し「GCC C Compiler」「すべてのオプション」を確認



コマンド:

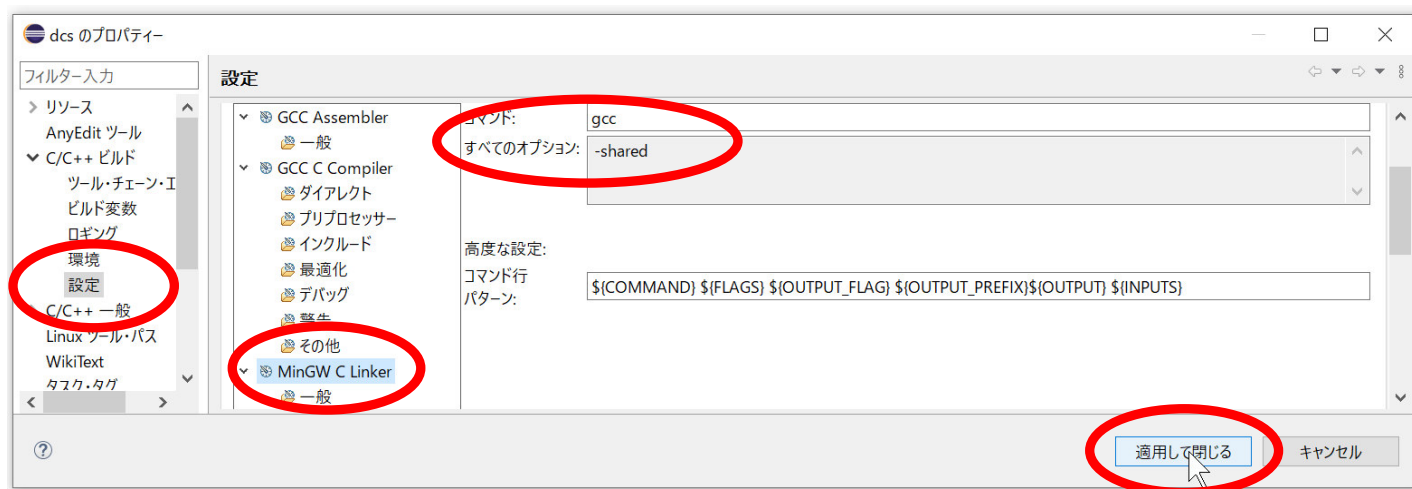
gcc

すべてのオプション:

```
-I"C:¥Users¥rabbit¥Documents¥dcs_build_src¥cpp¥dcs¥src" -I"C:¥Users¥rabbit
¥Documents¥dcs_build_src¥cpp¥jdc¥src" -I"C:¥ec2109cpp¥eclipse¥jre¥include"
-I"C:¥ec2109cpp¥eclipse¥ire¥include¥win32" -O3 -Wall -c -fmessage-length=0
```

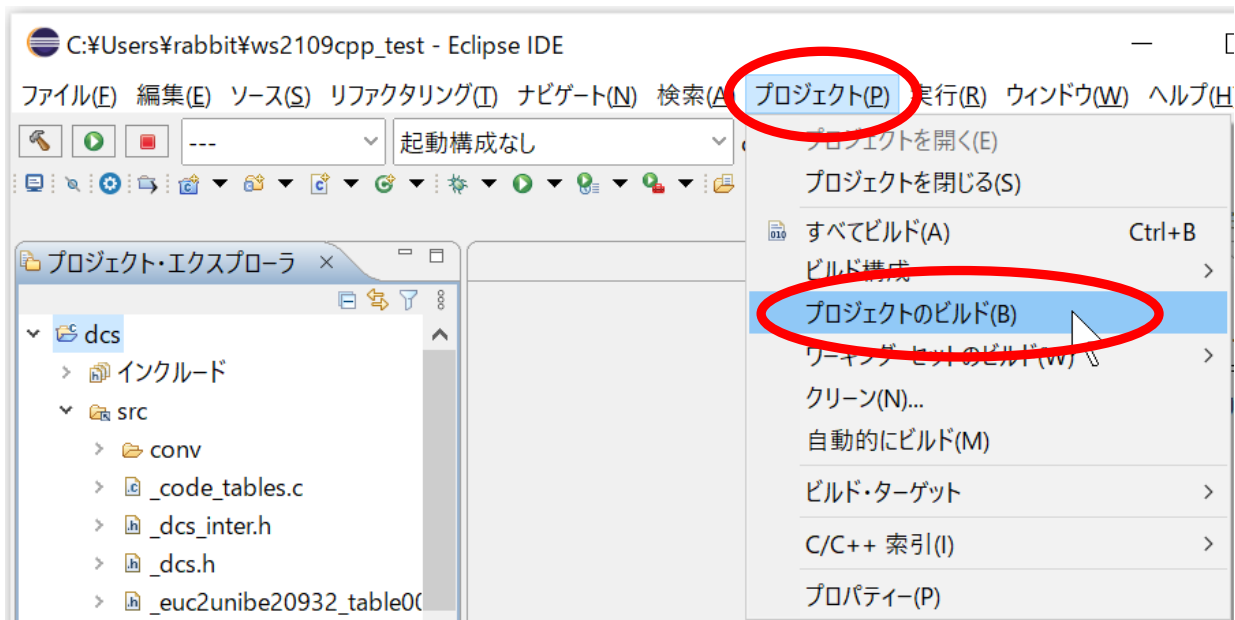
リンカー・オプションの確認

ダイアログより「C/C++ ビルド」「設定」を選択し「MinGW C Linker」「すべてのオプション」を確認し「適用して閉じる」

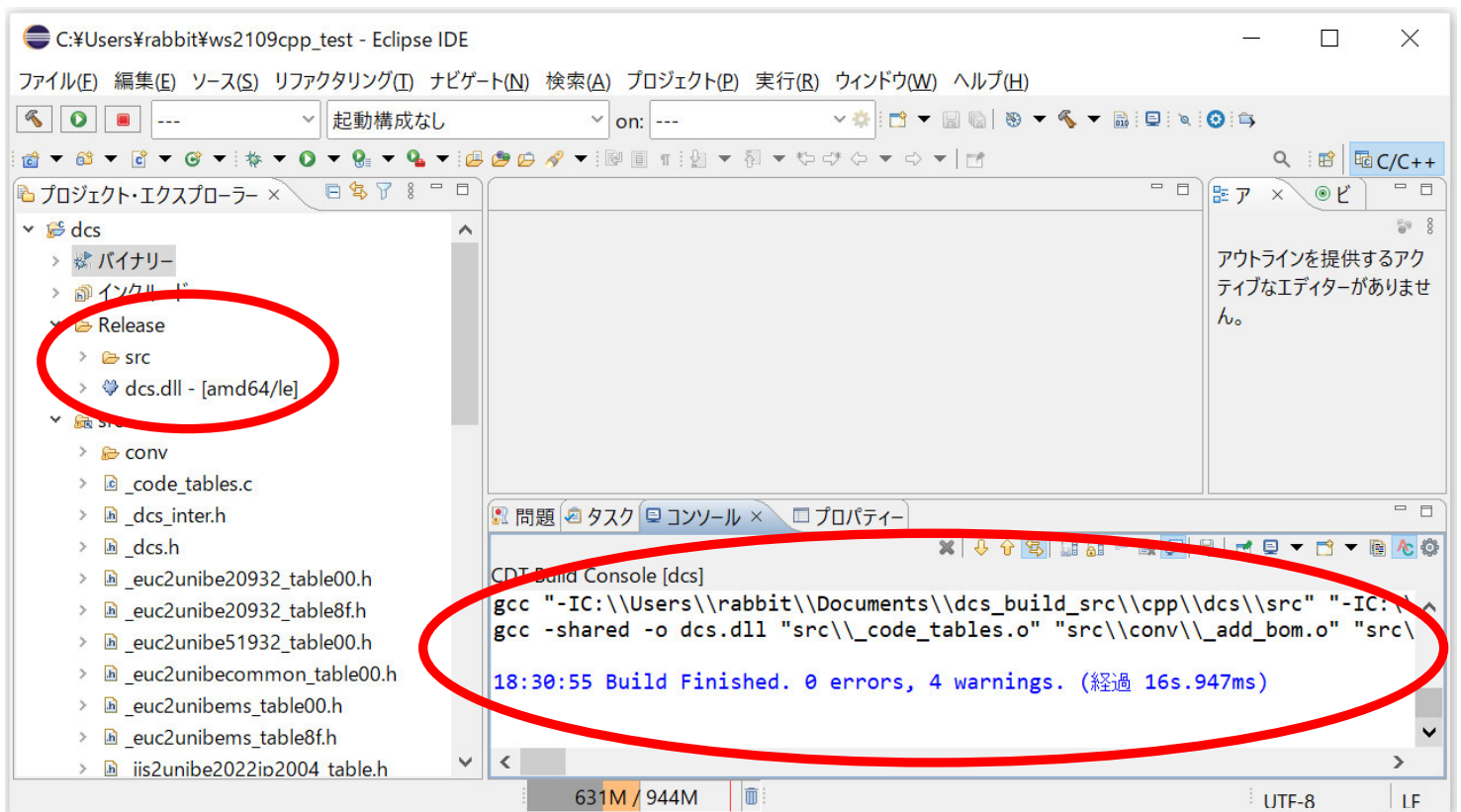


1-4. DCSプロジェクトのビルド

メニュー・バーより「プロジェクト」「プロジェクトのビルド」を選択



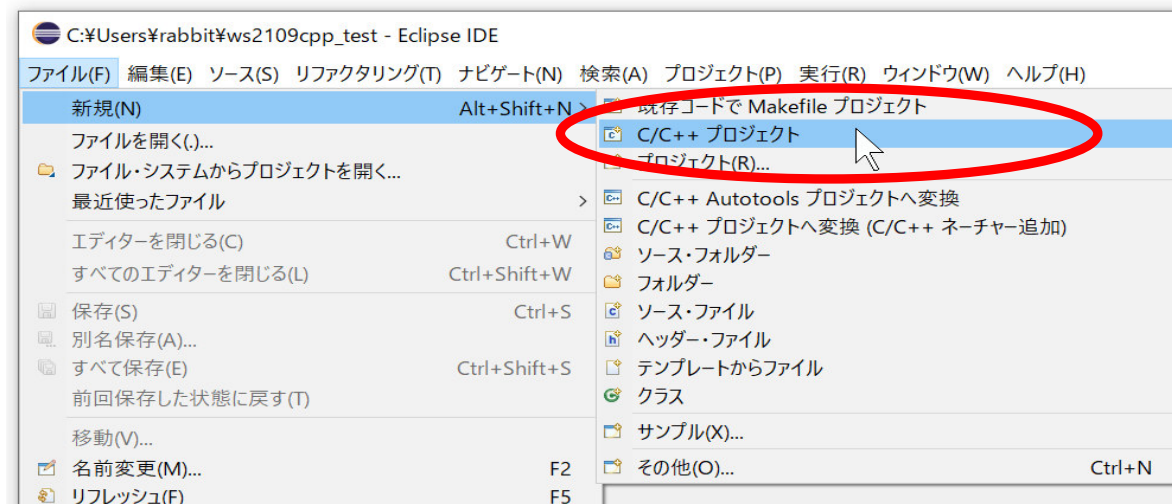
ビルドが正常に終了し「Release」にdcs.dllが生成される



2. DCSメイン(EXE)プロジェクト

2-1. DCSメイン・プロジェクト生成

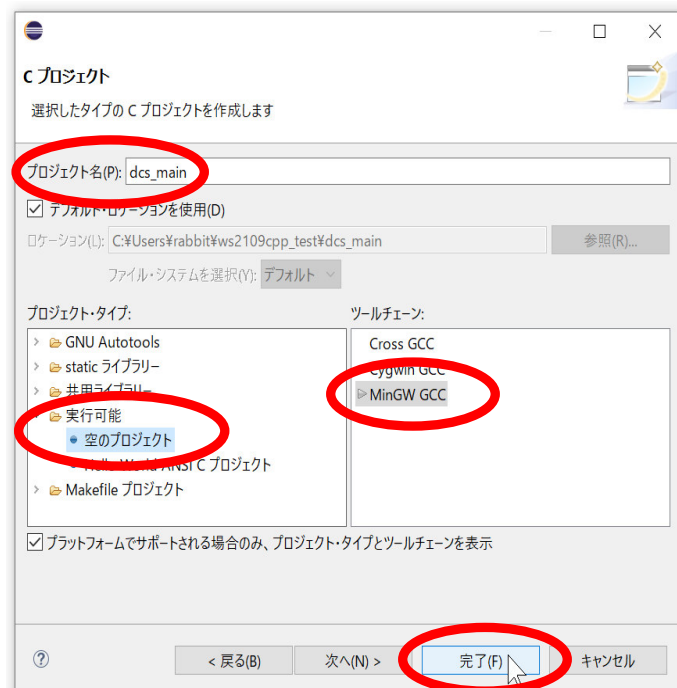
メニュー・バーより「ファイル」「新規」「C/C++ プロジェクト」を選択



ダイアログより「C 管理ビルド」を選択して「次へ」

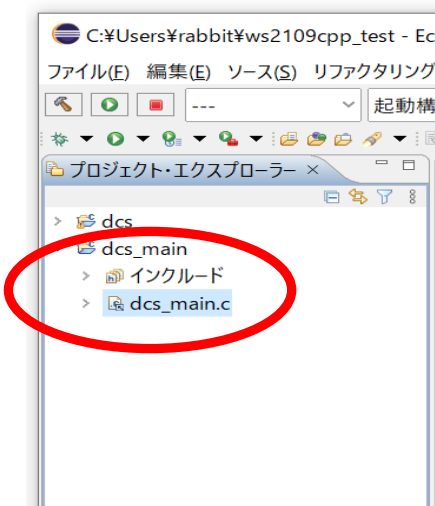


ダイアログより「プロジェクト名」を入力、「プロジェクト・タイプ」で「実行可能」「空のプロジェクト」を選択、「ツールチェーン」で「MinGW GCC」を選択し「完了」



2-2. DCSメイン・プロジェクトにソース・ファイル登録

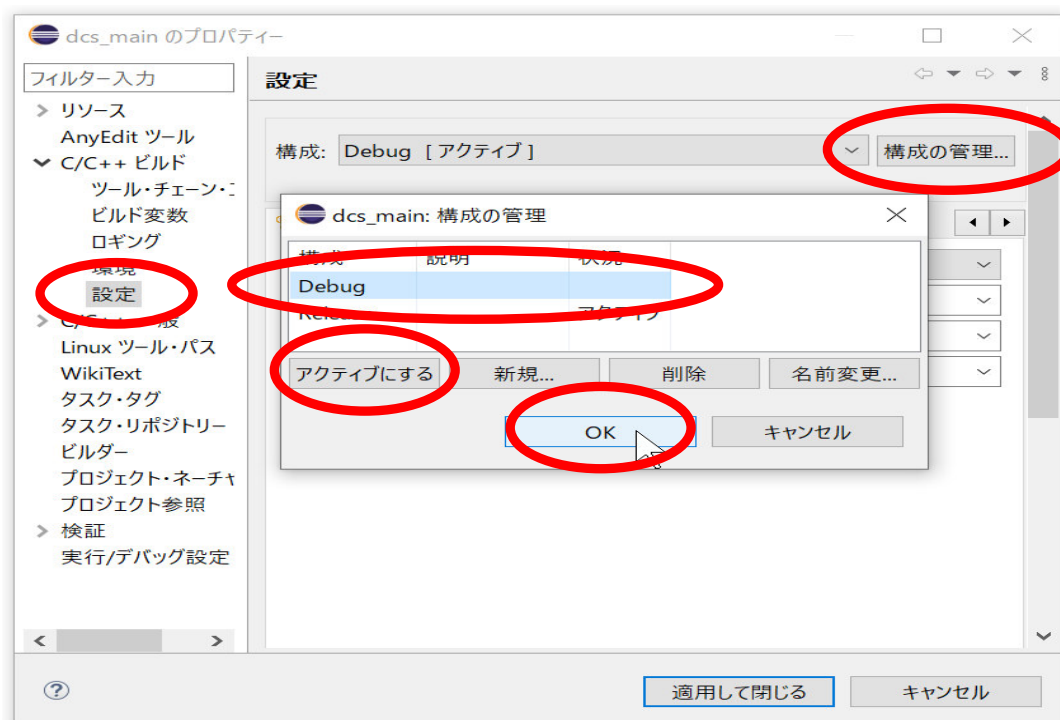
プロジェクト dcs_main が生成されるので dcs プロジェクトと同様にドラッグ&ドロップでソース・ファイルを登録する。ただし、dcs_main プロジェクトは dcs_main.c の1本のみなので dcs_main 直下に追加する。



2-3. DCSメイン・プロジェクトのプロパティ設定

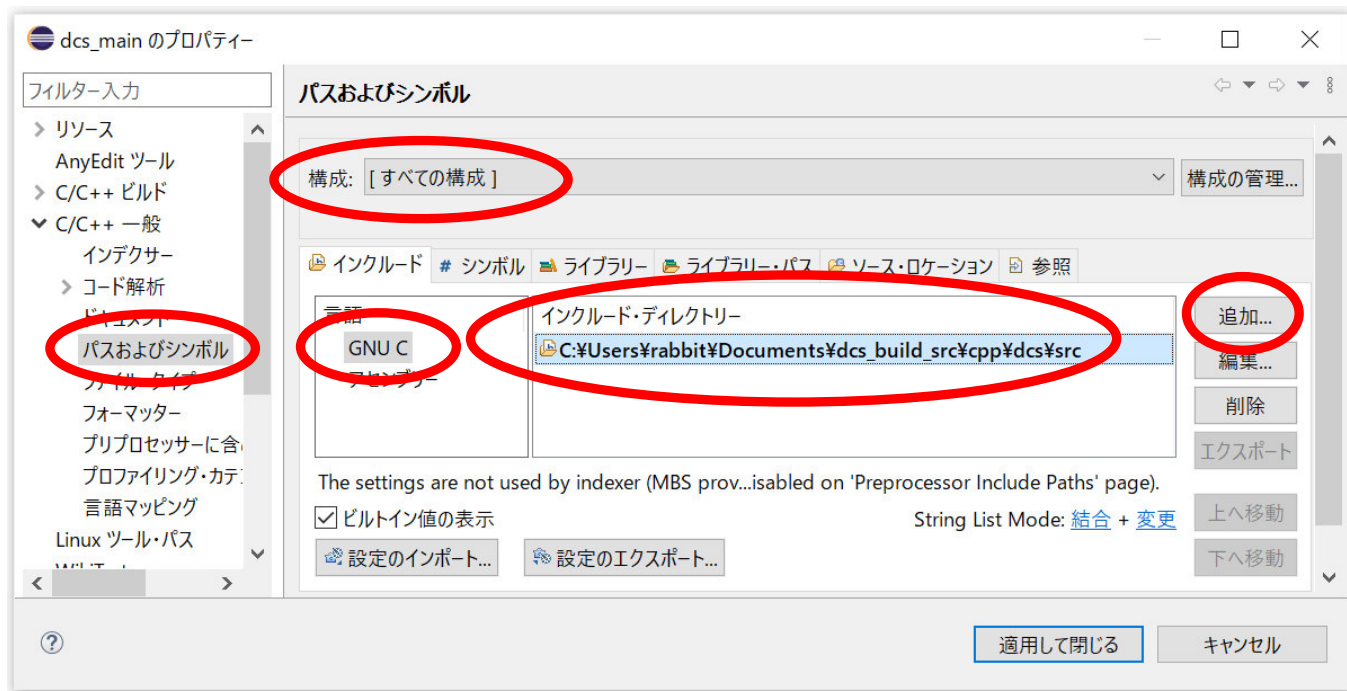
構成 Release のアクティブ化

ダイアログより「C/C++ ビルド」「設定」を選択し「構成の管理…」をクリックして「構成の管理」ダイアログより Release を選択し「アクティブにする」をクリックして「OK」



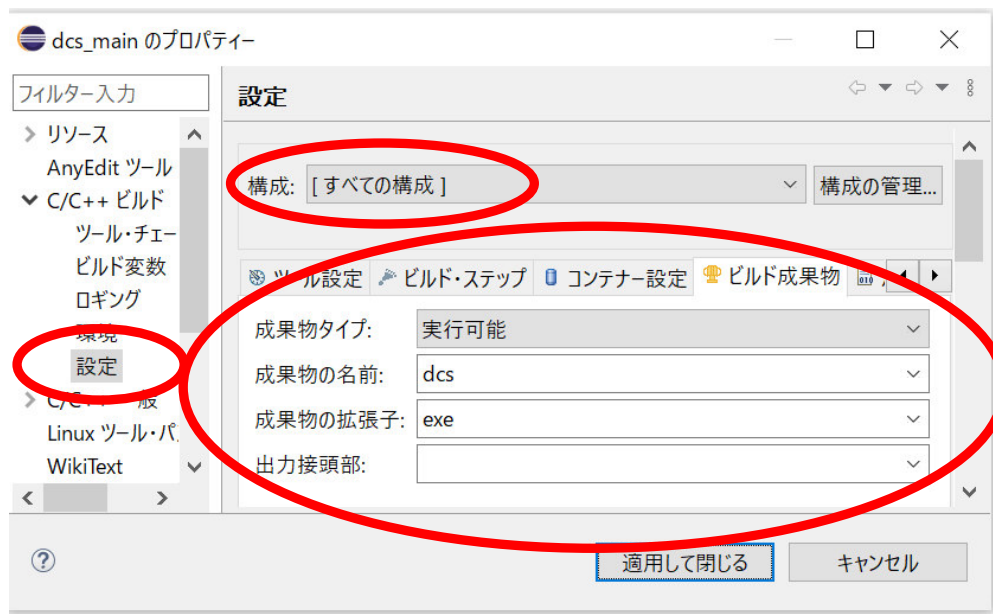
インクルード・パスの設定

ダイアログより「C/C++ 一般」「パスおよびシンボル」を選択し「構成」「すべての構成」を選択し「インクルード」タブより「言語」「GNU C」を選択し「インクルード・ディレクトリ」に「追加」ボタンをクリックして dcs の src パスを登録



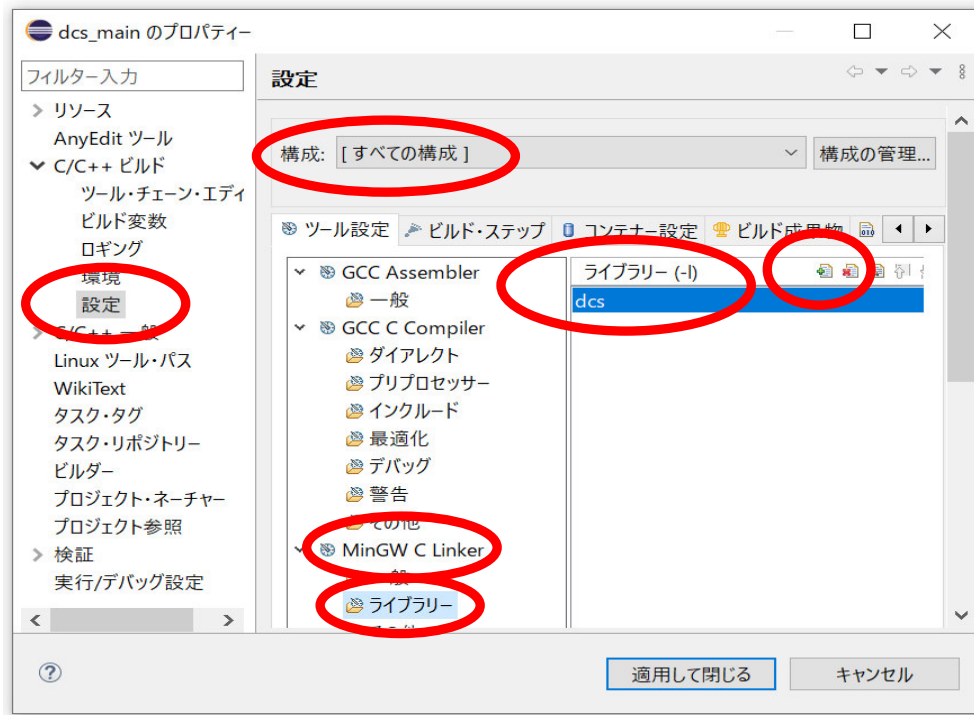
EXE名の設定

ダイアログより「C/C++ ビルド」「設定」を選択し「構成」「すべての構成」を選択し「ビルド成果物」タブより「成果物タイプ」「実行可能」、「成果物の名前」dcs、「成果物の拡張子」exe、「出力接頭部」を削除



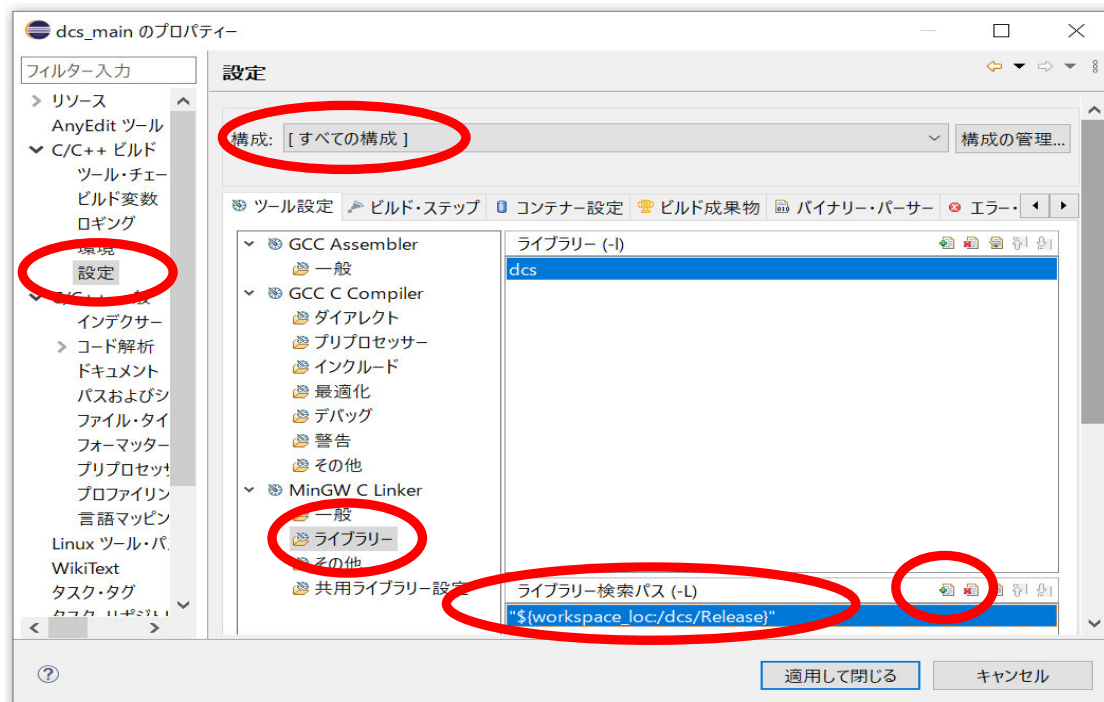
リンクDLL名の設定

ダイアログより「C/C++ ビルド」「設定」を選択し「構成」「すべての構成」を選択し「ツール設定」タブより「MinGW C Linker」「ライブラリー」を選択し右欄「ライブラリー」に「追加…」アイコンをクリックしてライブラリー名 dcs を登録



リンクDLLパスの設定

ダイアログより「C/C++ ビルド」「設定」を選択し「構成」「すべての構成」を選択し「ツール設定」タブより「MinGW C Linker」「ライブラリー」を選択し右欄「ライブラリー検索パス」「追加…」アイコンをクリックしてライブラリー検索パスを登録



コンパイラー・オプションの確認

ダイアログより「C/C++ ビルド」「設定」を選択し「GCC C Compiler」「すべてのオプション」を確認

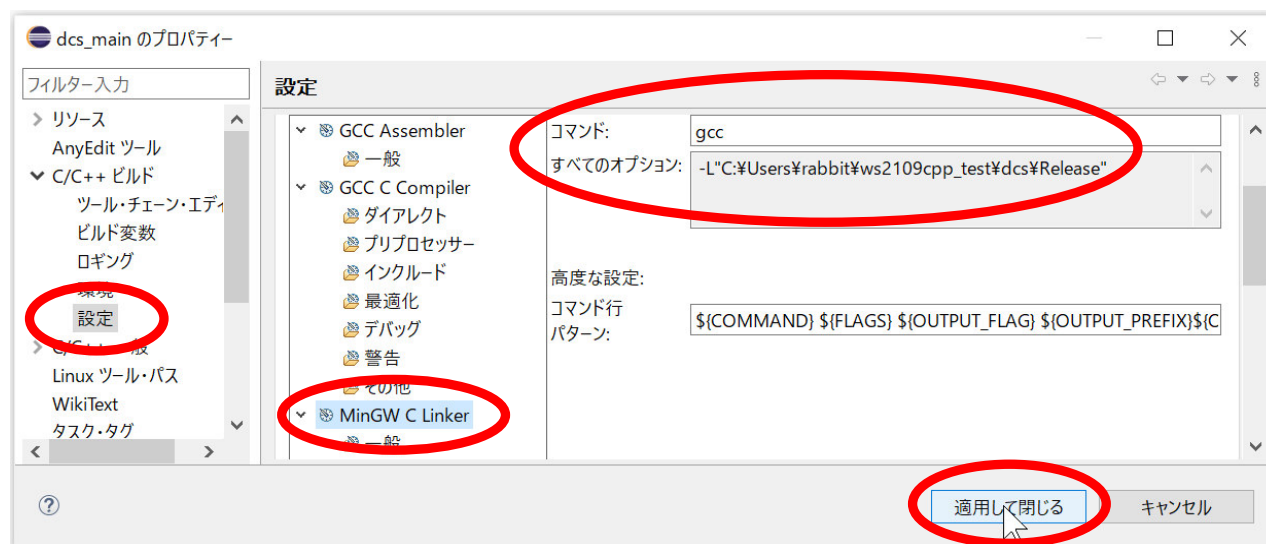


コマンド: gcc

すべてのオプション: -I"C:¥Users¥rabbit¥Documents¥dcs_build_src¥cpp¥dcs¥src" -O3 -Wall -c -fmessage-length=0

リンカー・オプションの確認

ダイアログより「C/C++ ビルド」「設定」を選択し「MinGW C Linker」「すべてのオプション」を確認し「適用して閉じる」

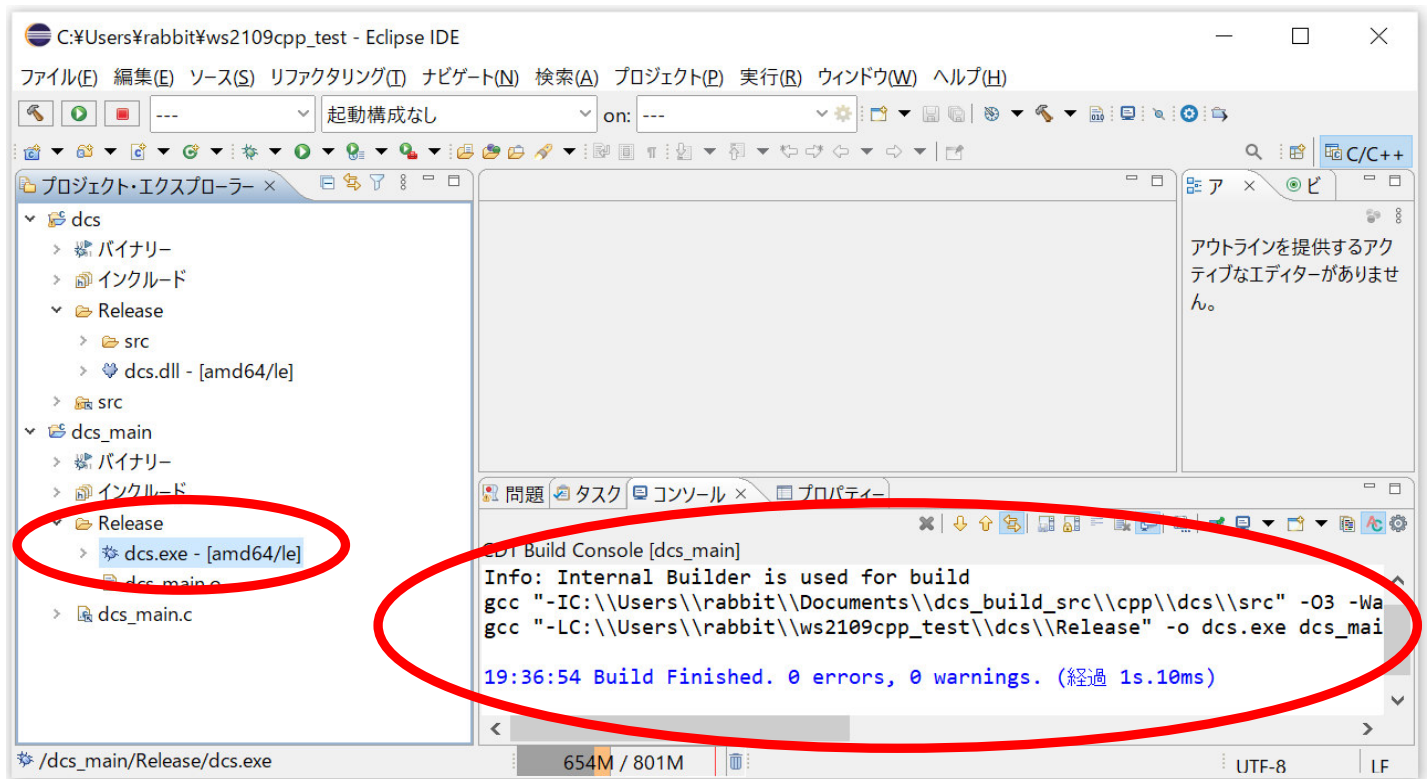


コマンド: gcc

すべてのオプション: -L"C:¥Users¥rabbit¥ws2109cpp_test¥dcs¥Release"

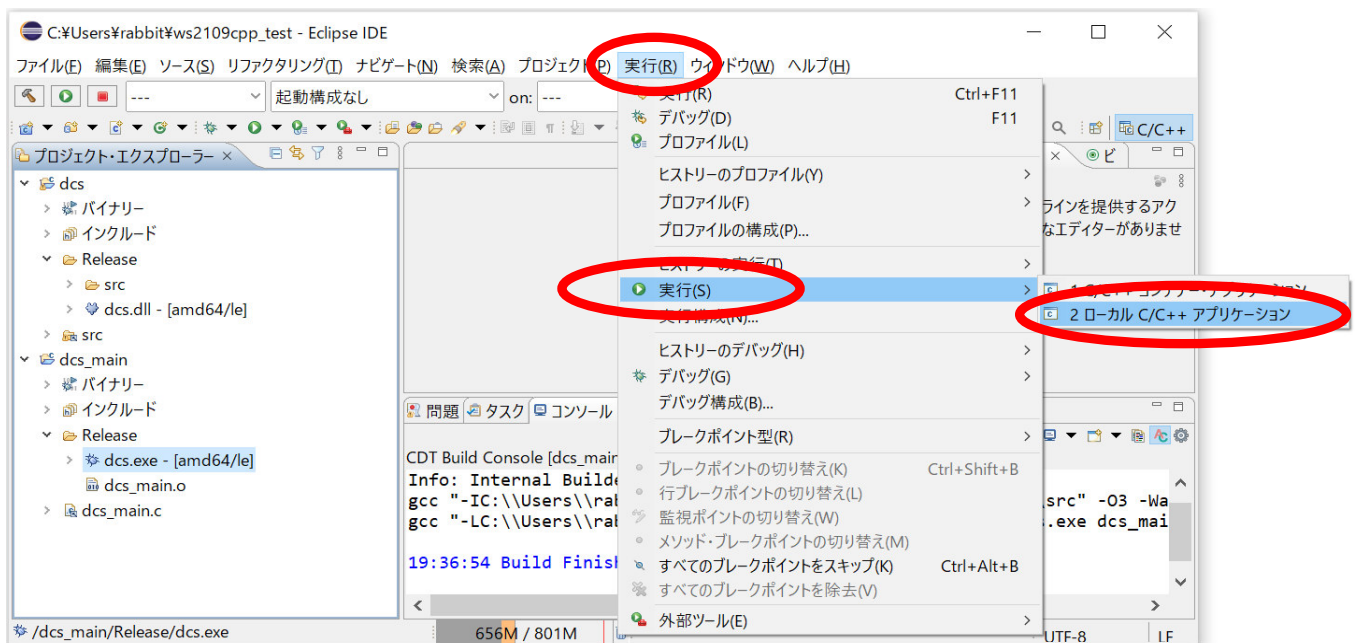
2-4. DCSメイン・プロジェクトのビルド

プロジェクト・エクスプローラーの dcs_main を選択し、メニュー・バーより「プロジェクト」「プロジェクトのビルド」を選択すると、ビルドが正常に終了し「Release」に dcs.exe が生成される

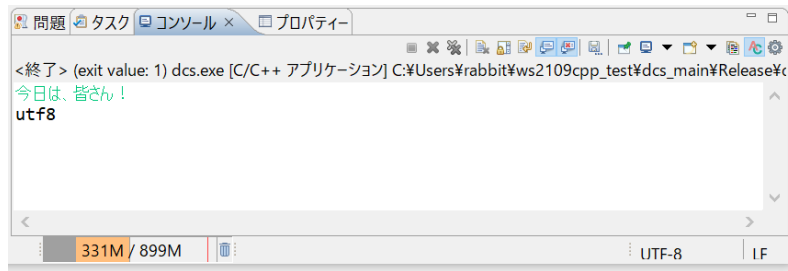


2-5. DCSメインの実行

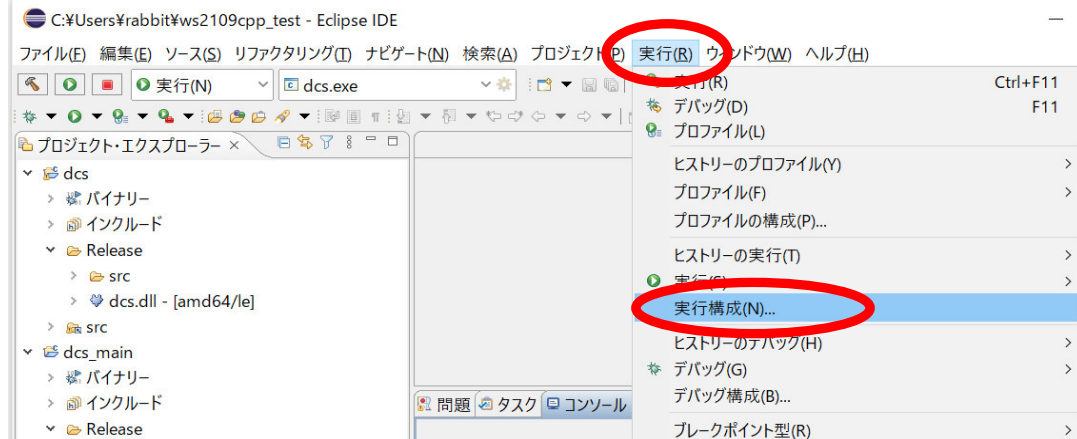
プロジェクト・エクスプローラーの dcs_main を選択し、メニュー・バーより「実行」「実行 (S)」「2 ローカル C/C++ アプリケーション」を選択



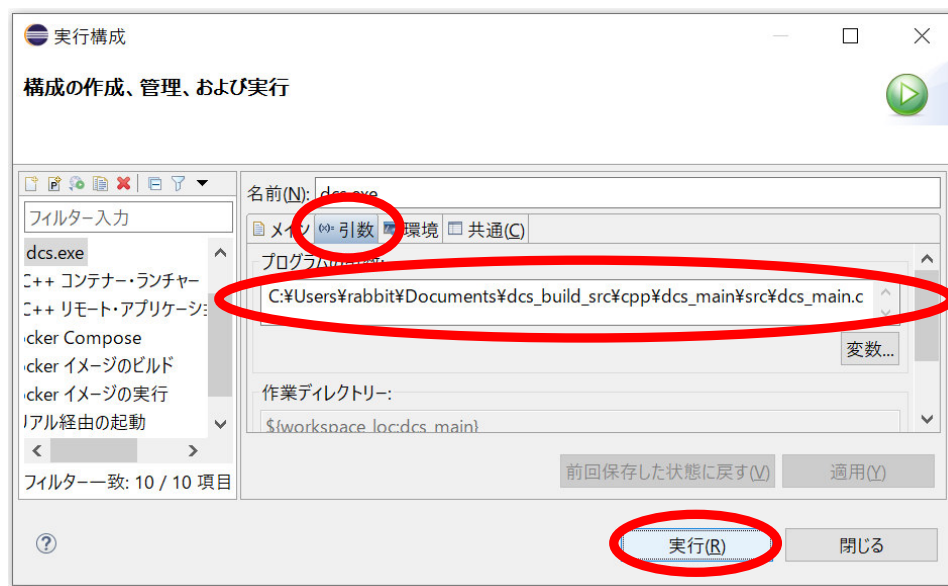
コンソールからのキー入力になるため、適当に入力して Enter キー後、Ctrl+Z



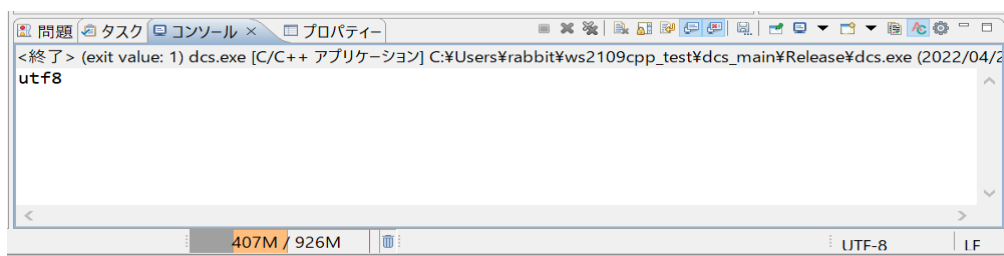
一度実行を行なうと実行構成が生成されるので、メニュー・バーより「実行」「実行構成」を選択



「引数」を選択し、「プログラムの引数」に入力ファイル・パスを指定して再実行



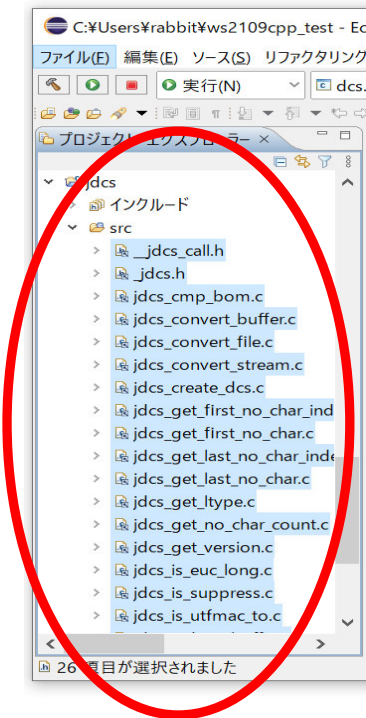
入力ファイルがUTF-8 と正しく判定される



3. JDCS(DLL)プロジェクト

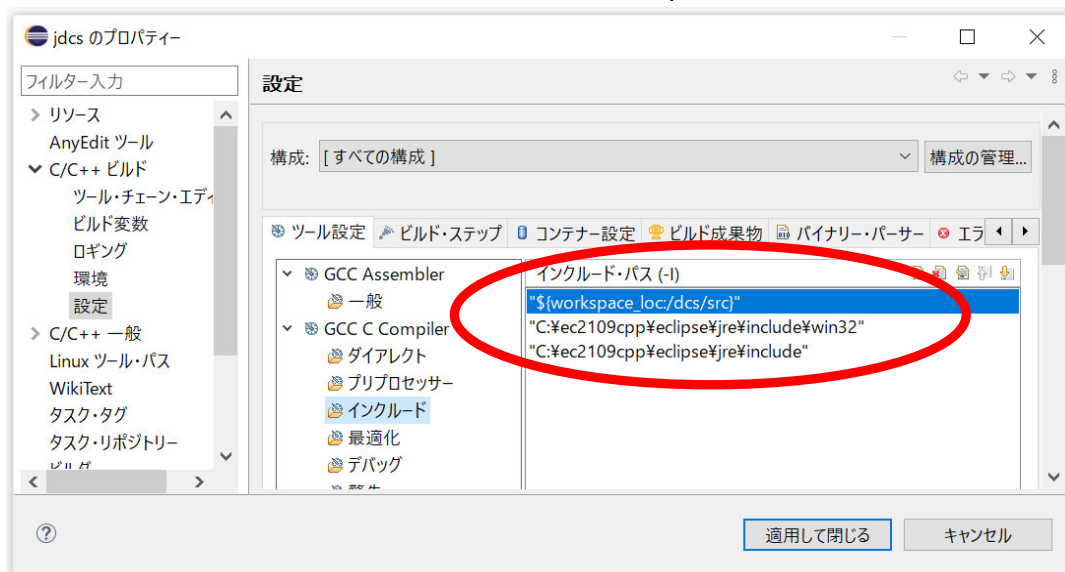
3-1. JDCSプロジェクト生成

DCSプロジェクトと同様にJDCSプロジェクトを生成し、プロジェクト・フォルダー jdcс に jdcс のソース・フォルダー src をドラッグ&ドロップにより登録



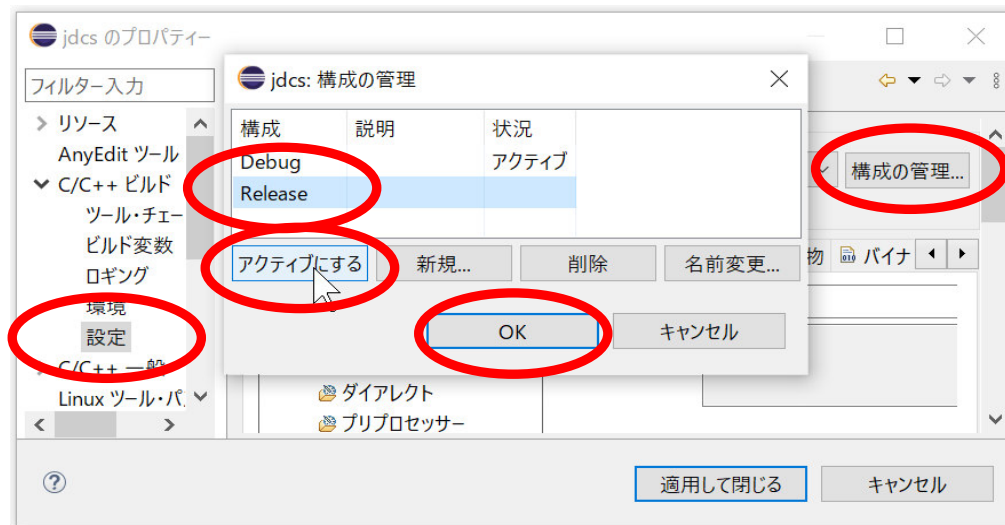
3-2. JDCSプロジェクトのプロパティ設定

インクルード・パスには dcsの srcと JRE配下にあるJNI用の2パスを設定。JREのバージョンは1.1以上であれば可。ここではEclipseのものを使用

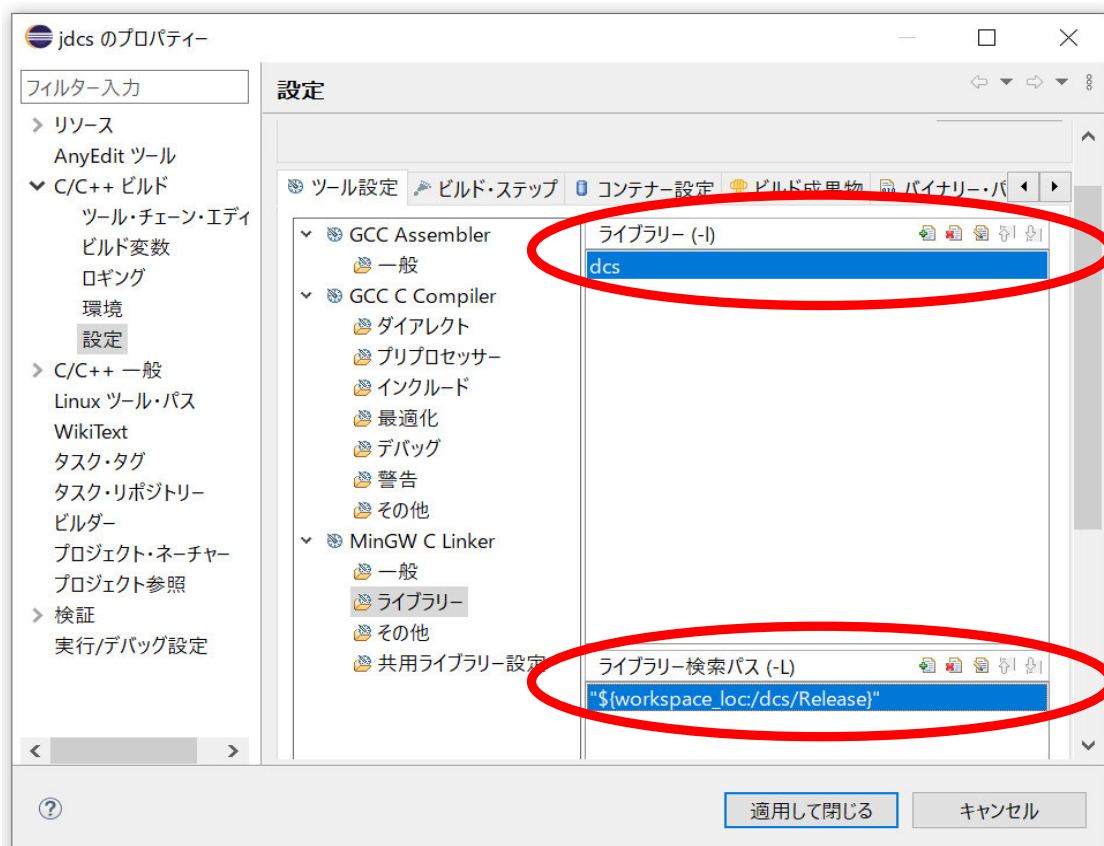


構成 Release のアクティブ化

ダイアログより「C/C++ ビルド」「設定」を選択し「構成の管理…」をクリックして「構成の管理」アイアログより Release を選択し「アクティブにする」をクリックして「OK」

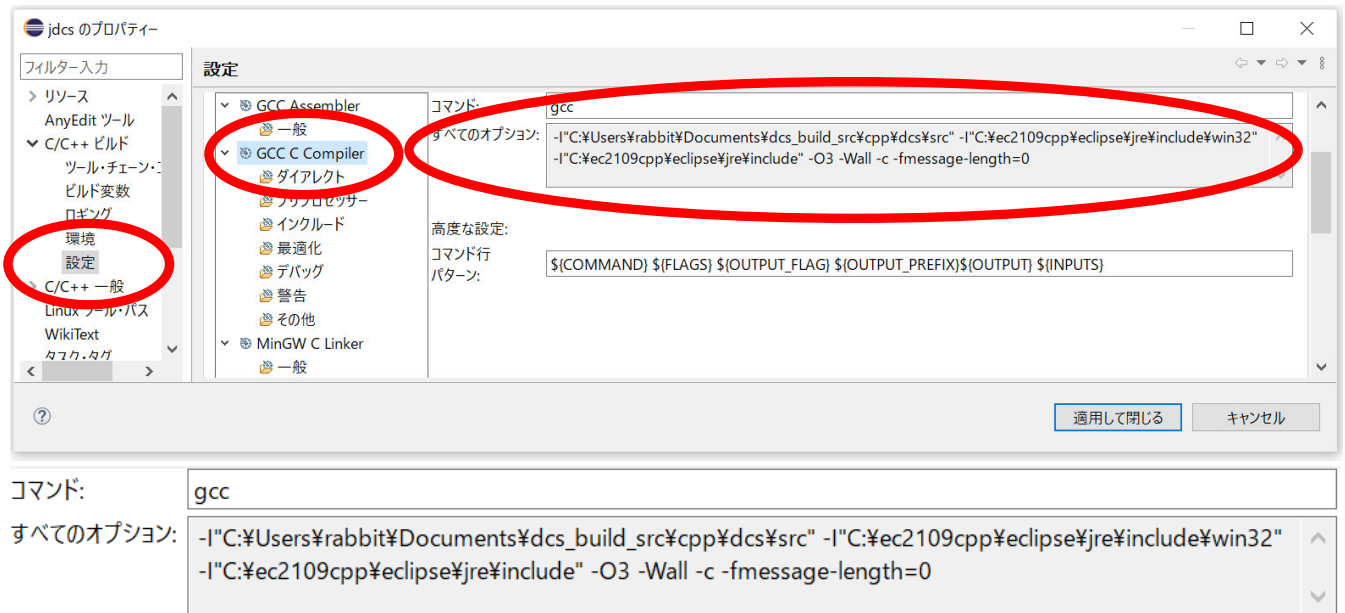


ライブラリーとライブラリー検索パスには dcs を設定する



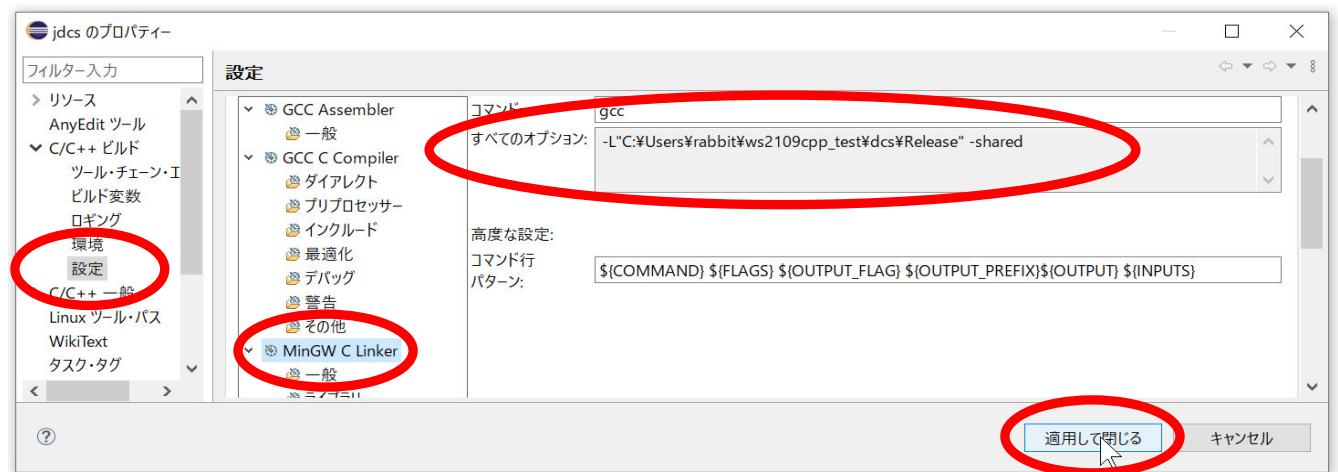
コンパイラー・オプションの確認

ダイアログより「C/C++ ビルド」「設定」を選択し「GCC C Compiler」「すべてのオプション」を確認



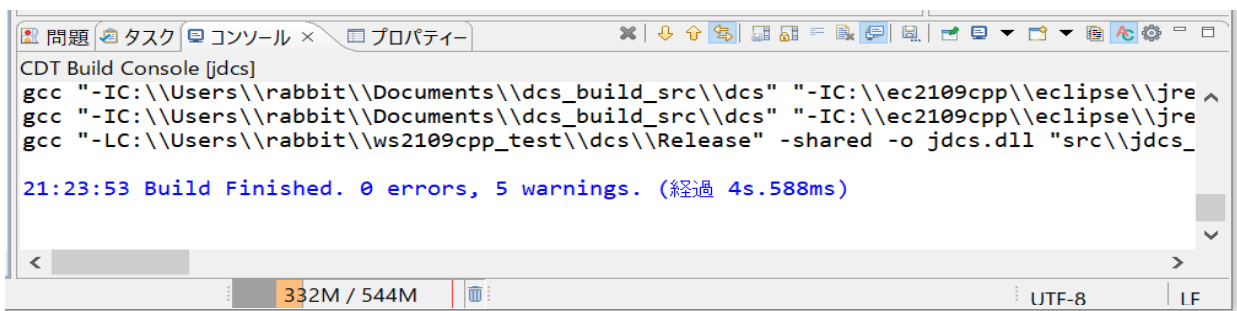
リンカー・オプションの確認

ダイアログより「C/C++ ビルド」「設定」を選択し「MinGW C Linker」「すべてのオプション」を確認し「適用して閉じる」



3-3. JDCSプロジェクトのビルド

メニュー・バーより「プロジェクト」「プロジェクトのビルド」を選択すると、ビルドが正常に終了し「Release」にjdc's.dll が生成される



4. JDCS(JAR)プロジェクト

4-1. JDCSプロジェクト生成

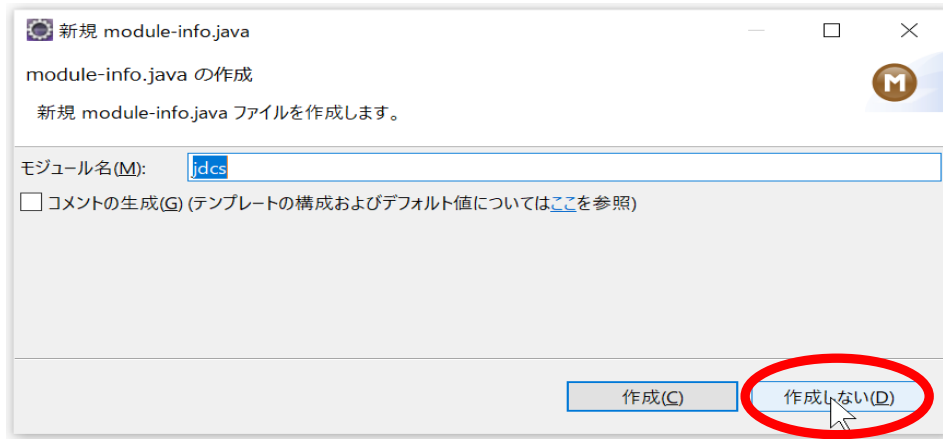
メニュー・バーより「ファイル」「新規」「Java プロジェクト」を選択



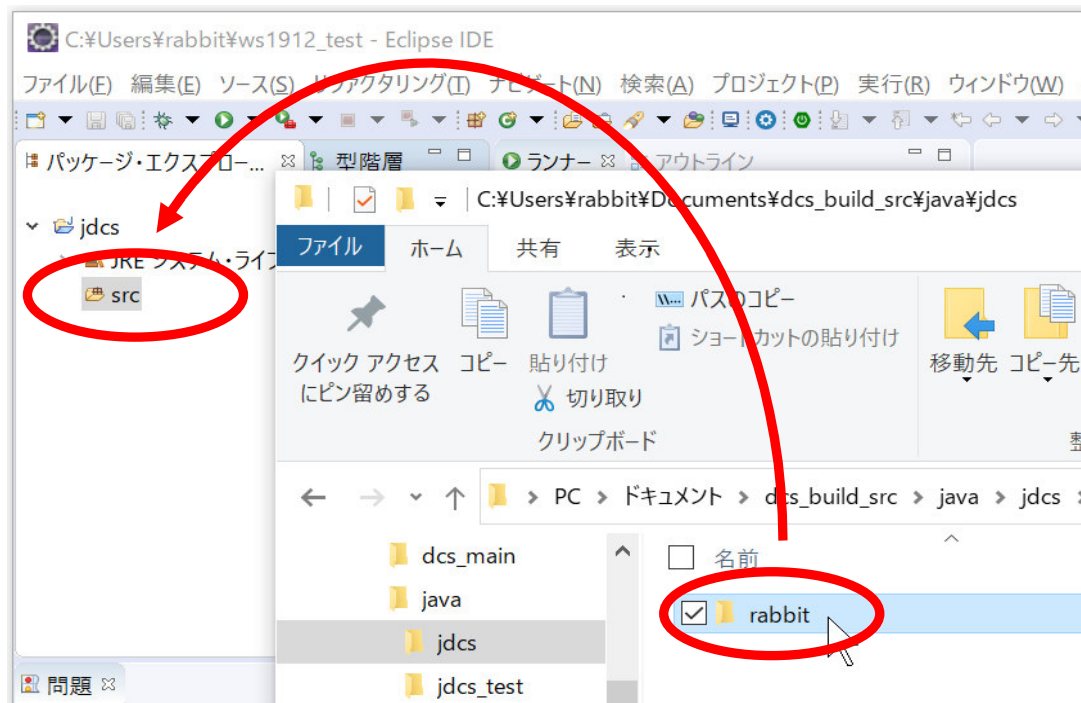
プロジェクト名を入力、JREバージョン 11 を設定して完了



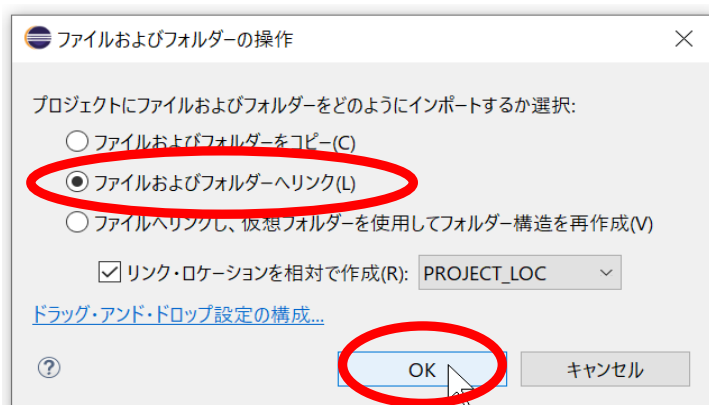
module-info.java は作成しない



パッケージの先頭 rabbit をプロジェクトの src へドラッグ&ドロップ

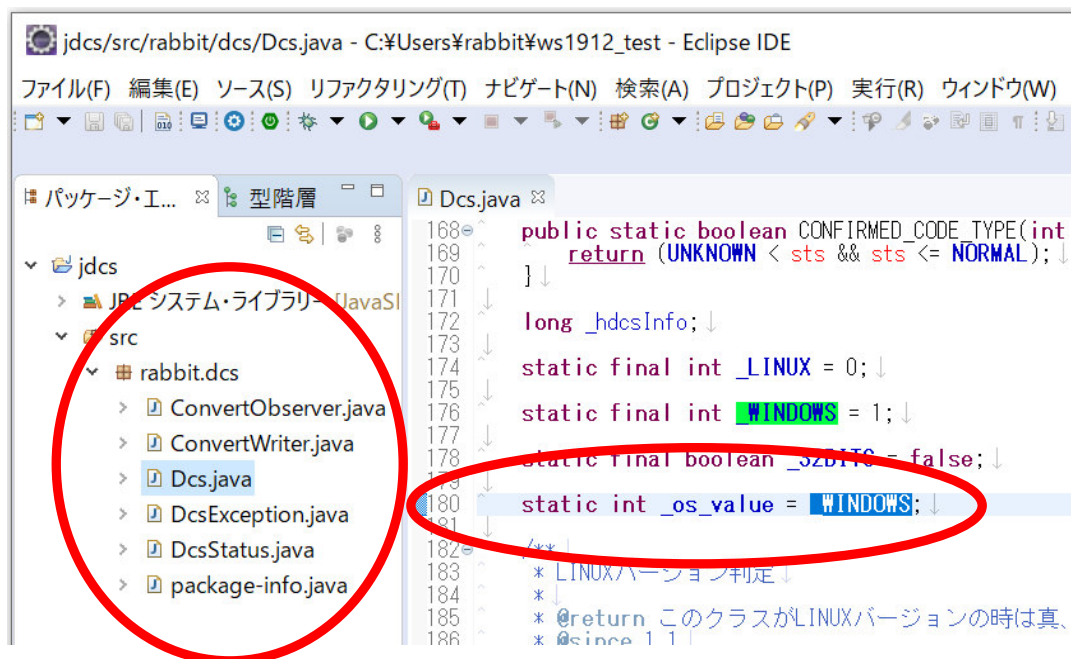


ダイアログより「ファイルおよびフォルダーへリンク」を選択（他の選択肢も可）して「OK」

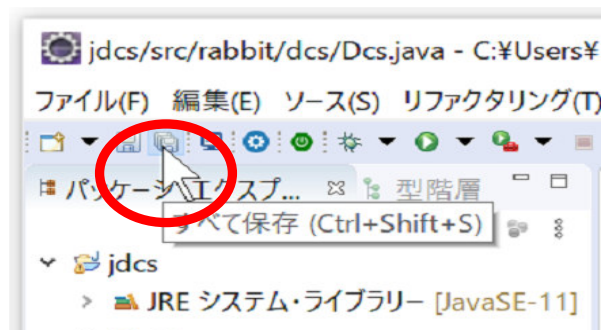


4-2. JDCSプロジェクト・ビルド

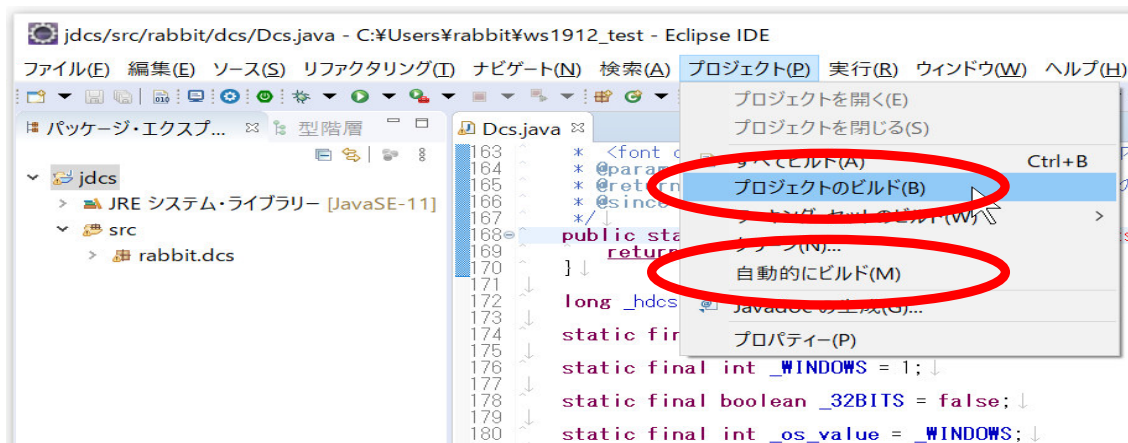
ソース・ファイルが登録されたことを確認し、**Windows** なら、**Dcs.java** の **_os_value** を **_WINDOWS** に設定 (180行)、**Linux** ならば **_LINUX** に設定



ソース・ファイルを編集したら「すべて保存」

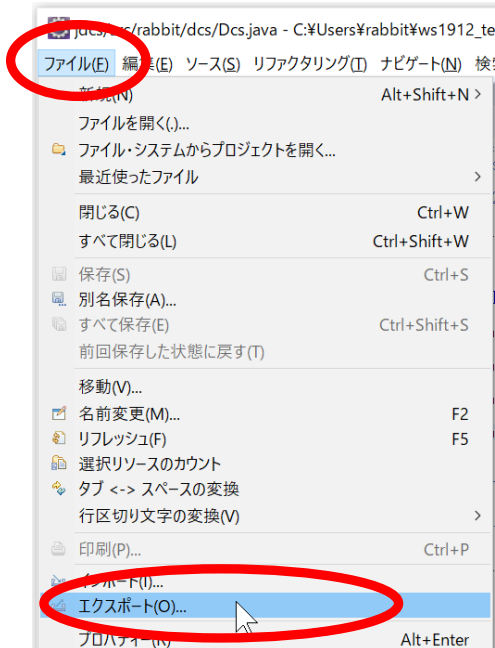


「自動的にビルド」が選択されていない場合は「プロジェクトのビルド」

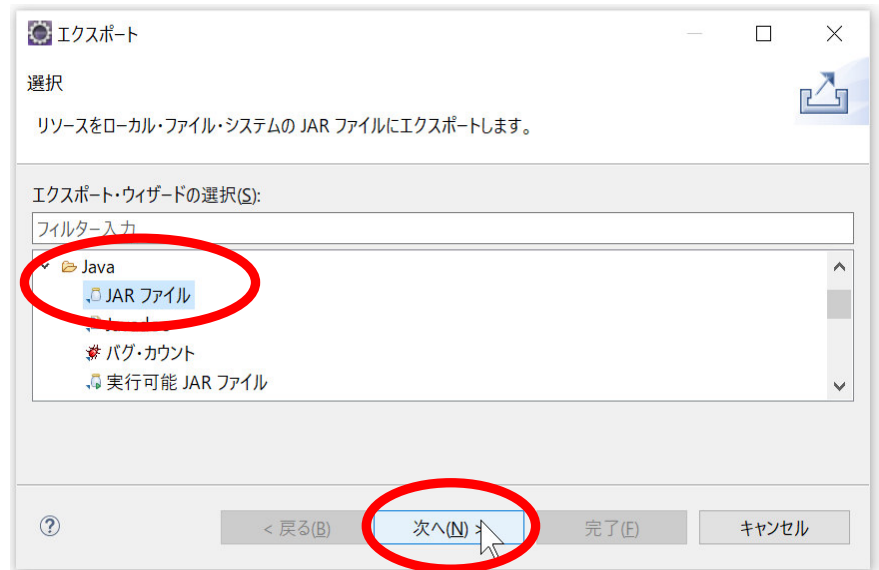


4 - 3. JDCSプロジェクト JAR 生成

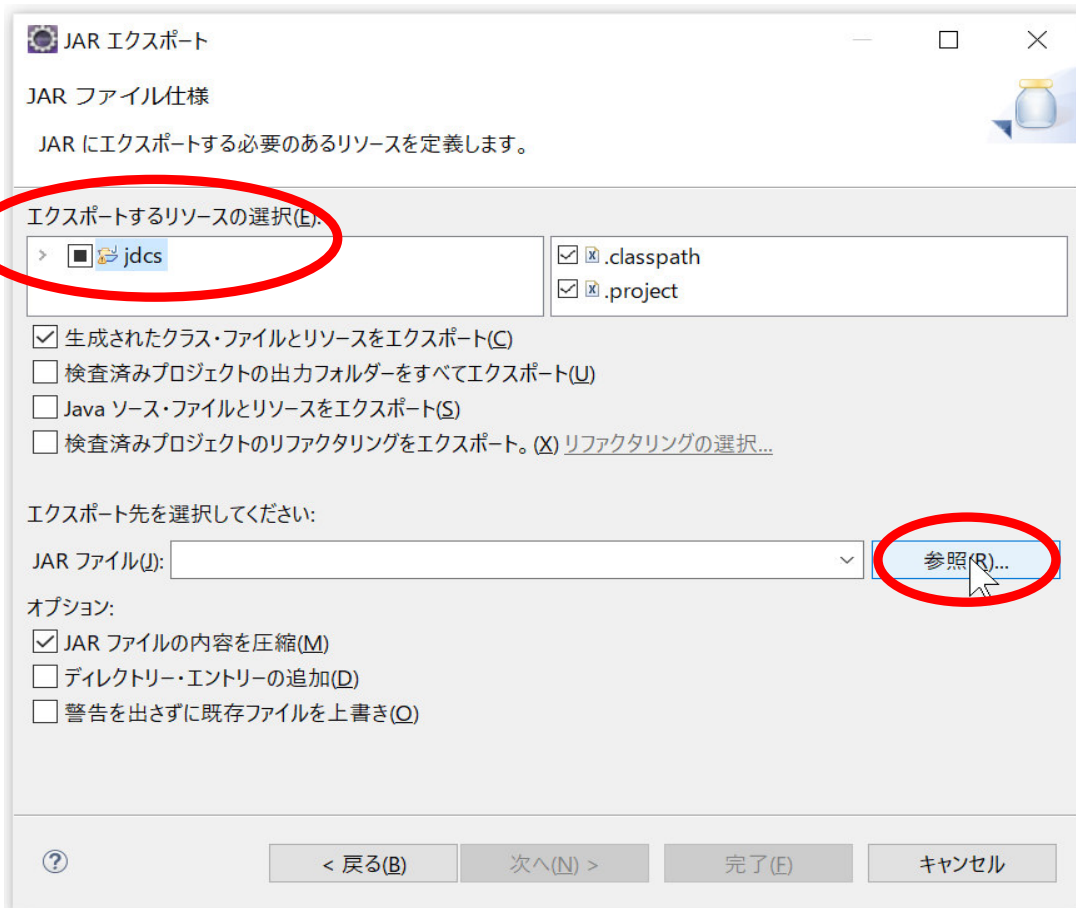
「ファイル」「エクスポート」を選択



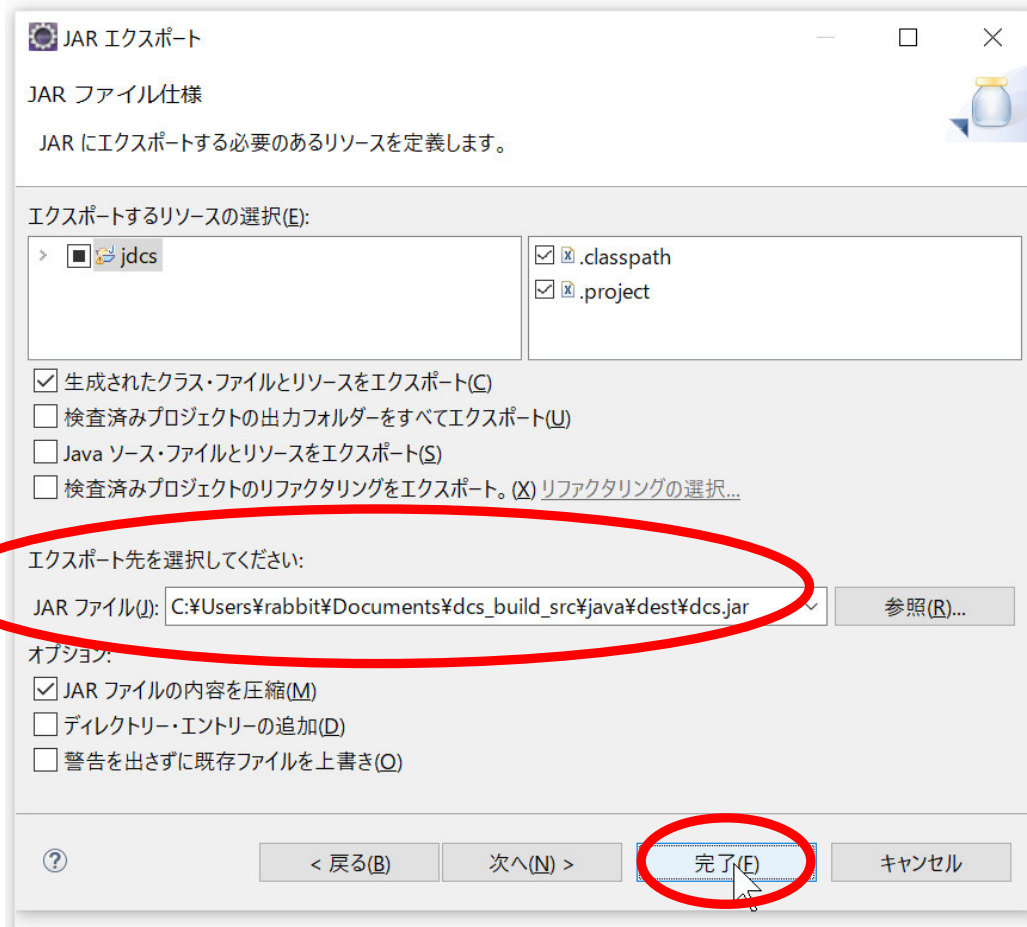
ダイアログより「Java」「JAR ファイル」を選択して「次へ」



正しく jdcс が選択されていることを確認しエクスポート先の「参照」をクリックし、ダイアログから JAR ファイルを設定



エクスポート先のパスを確認して「完了」をクリック

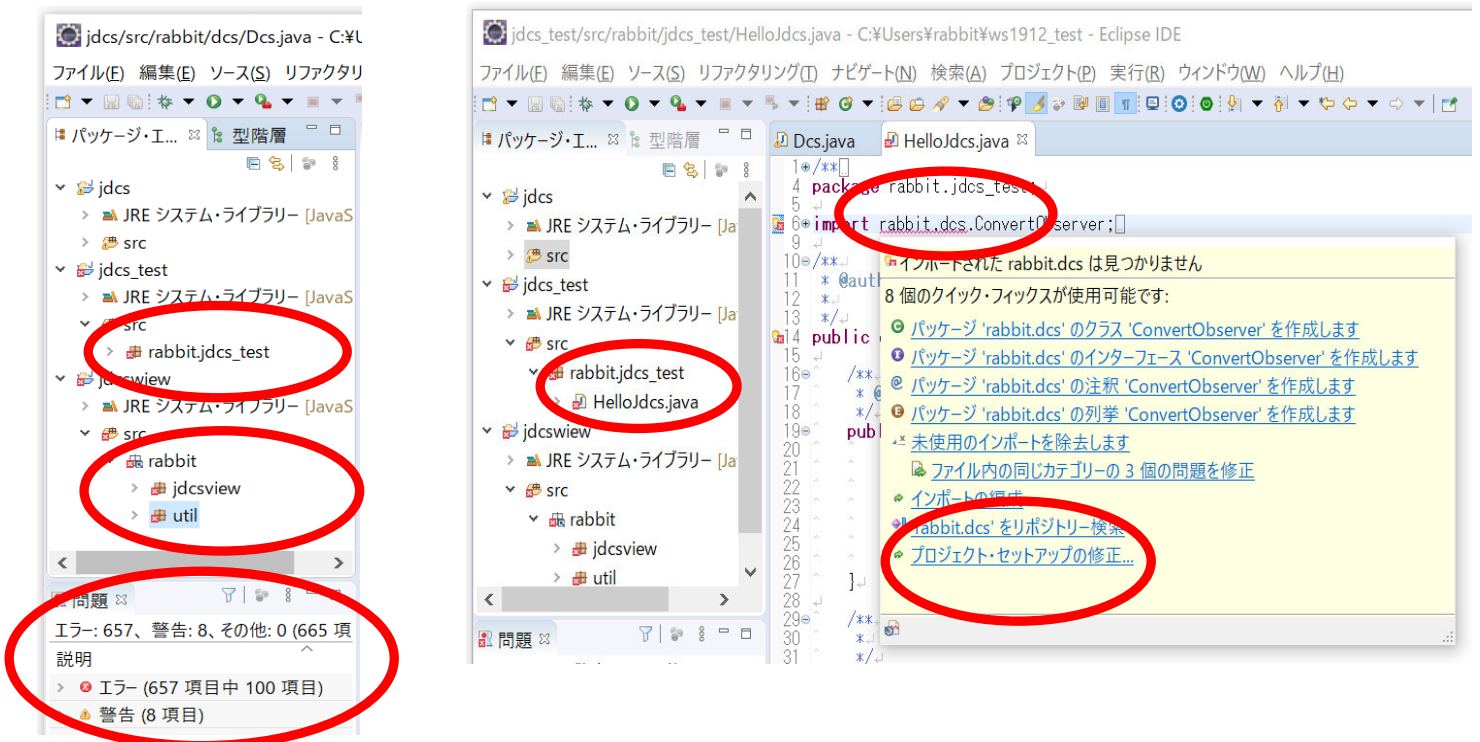


5. DCSビューアー/JDCSテスト・プロジェクト

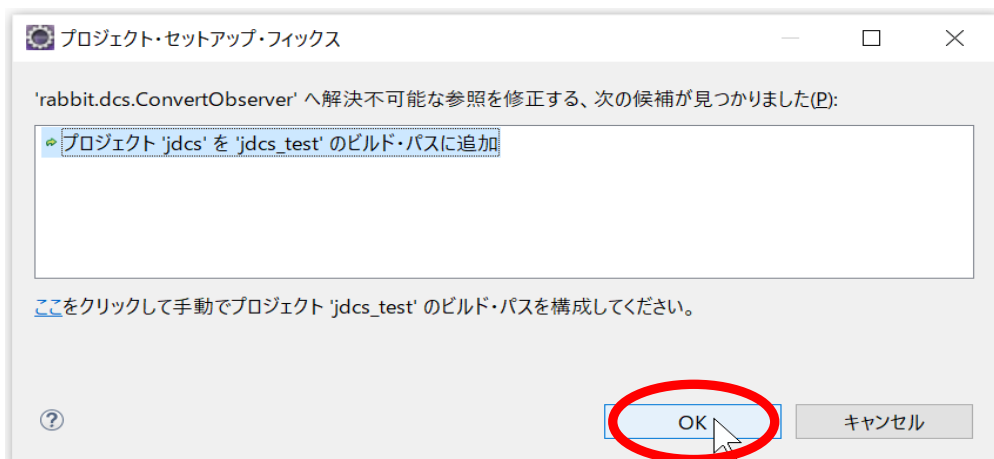
5 - 1. DCSビューアー/JDCSテスト・プロジェクト・ビルド

jdcs_test, jdcsview も同様にプロジェクト生成を行なうが、**未解決の参照プロジェクトや参照JARファイルが有るためエラーが発生する**

プロジェクト jdcs_test にある HelloJdcs.java を開いてエラーとなっている rabbit.dcs にマウス・カーソルを合わせて、表示されるクイック・フィックス「プロジェクト・セットアップ」の修正をクリック



jdcs をビルドパスに追加「OK」これでプロジェクト jdcs_test のエラーは解消する

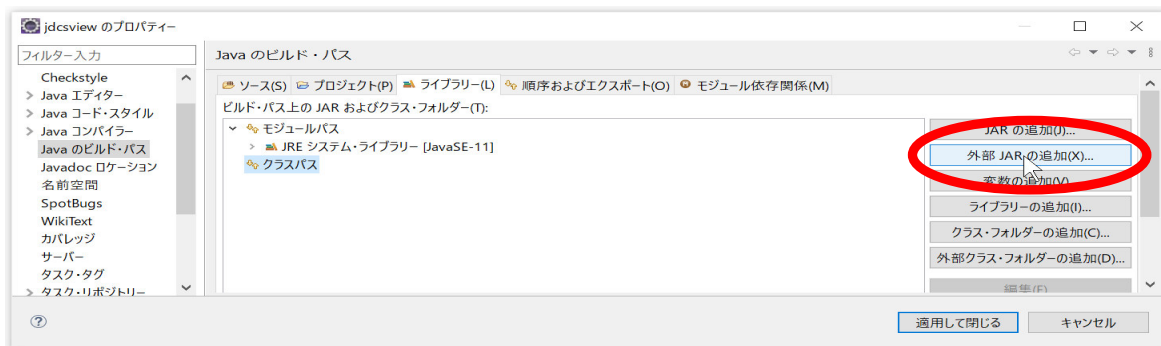


5 - 2. DCSビューア用 JAR ファイルの登録

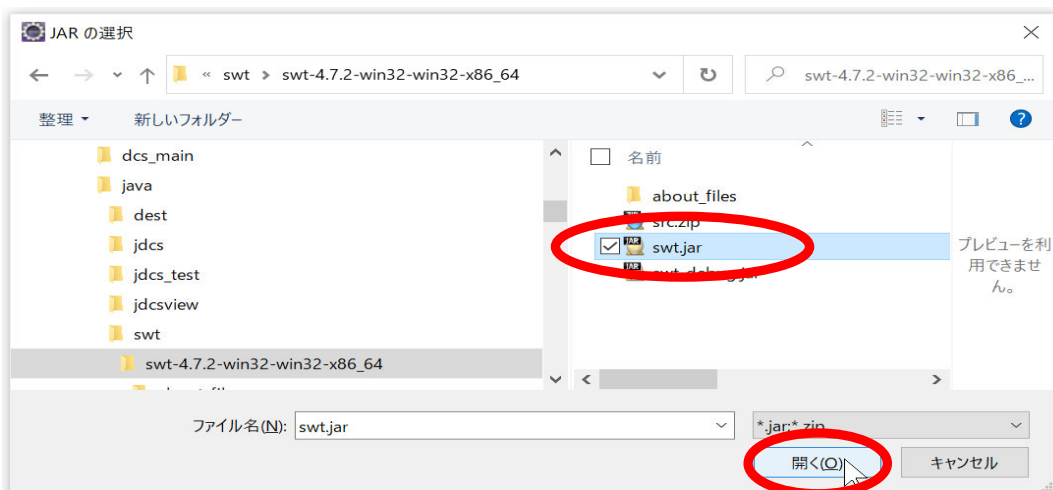
jdcsviewer も同様に jdcs を登録するが、他に SWT, JFace の 2 JAR ファイルを用意、登録する。ZIP などのアーカイブ・ファイルとなっている場合は解凍する。ただし **JFace は Windows 上で disable になっている Control の ツール・チップを表示するために、GridDataFactory を JDcsView クラスで使用しているだけなので、必要なければ import 文を含めた 6 箇所を削除。** そうすれば JFace の登録は不要となる



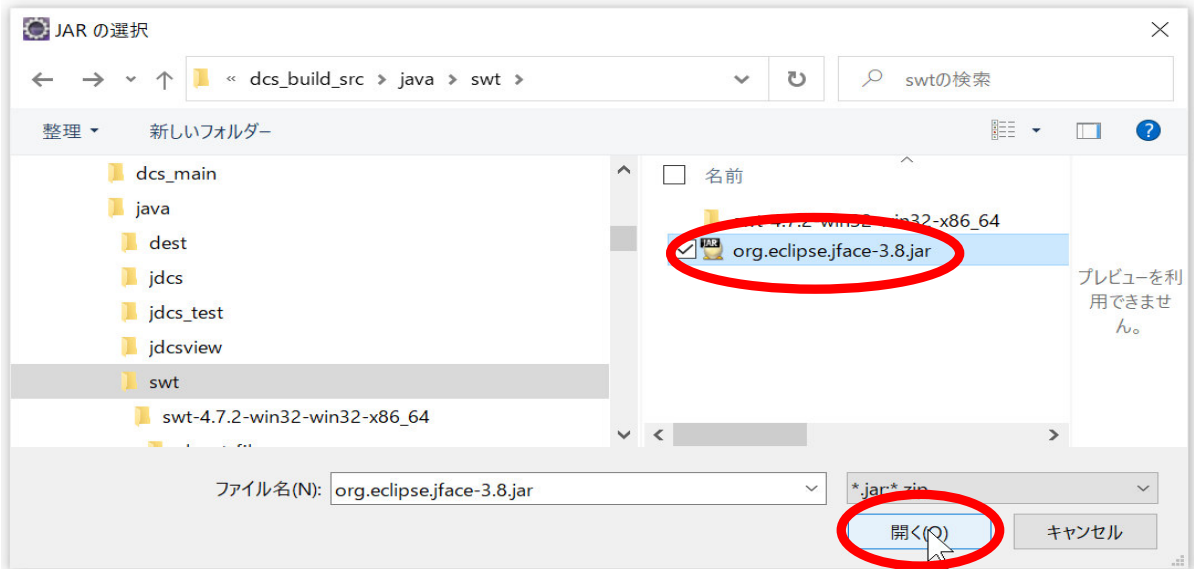
jdcsviewer のプロパティから「Java のビルド・パス」「ライブラリー」「クラスパス」を選択し「外部 JAR の追加」をクリック



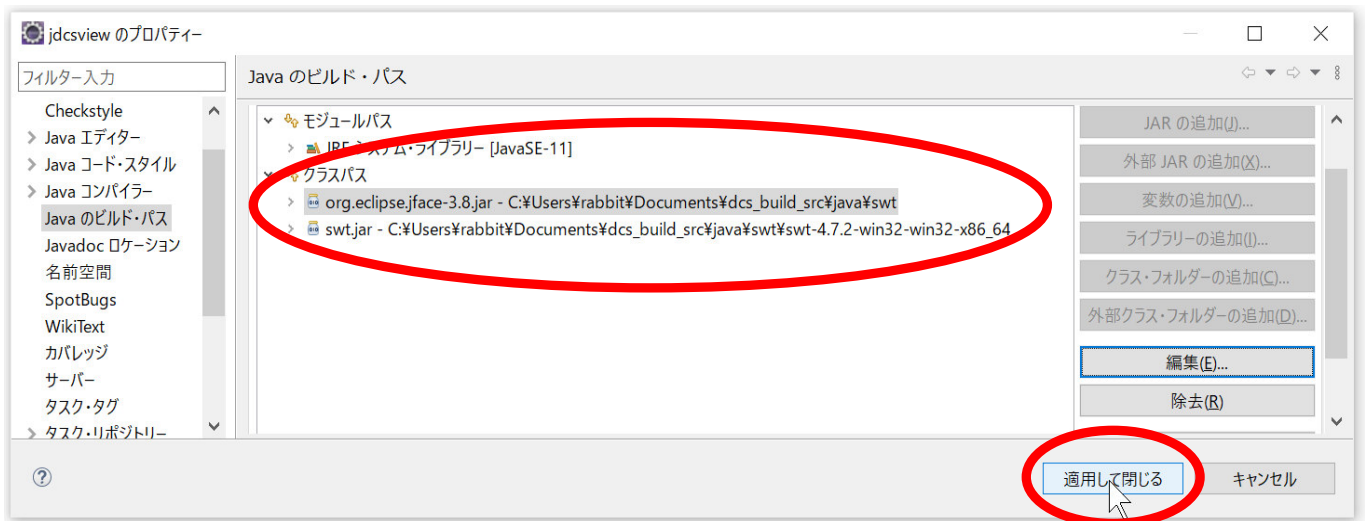
用意しておいた SWT, JFace の JAR ファイルを選択。まずは SWT から



続いて JFace の JAR ファイルを選択

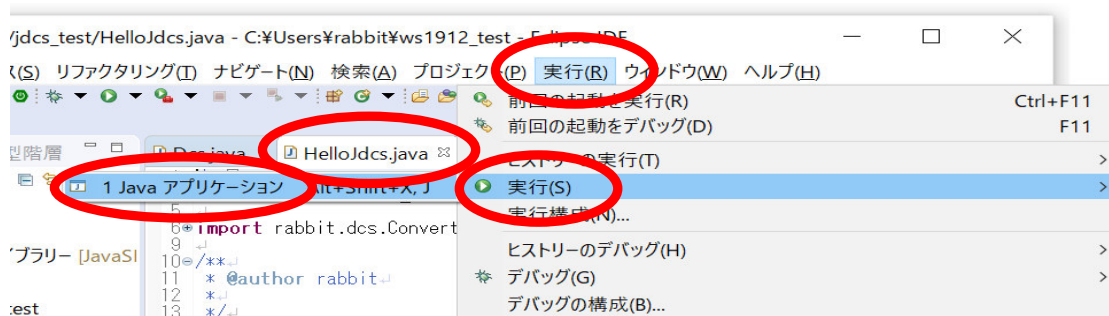


SWT, JFACE の 2 JAR ファイルが登録されたことを確認して「適用して閉じる」
これで jdcsview のエラーも解消する

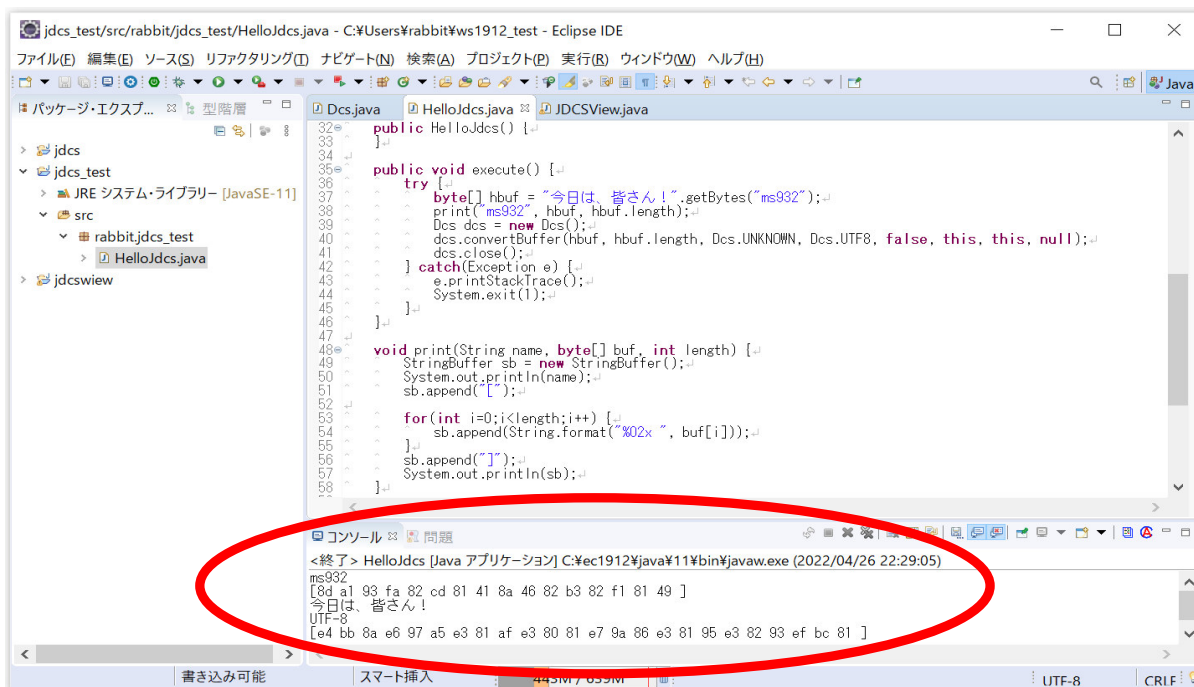


5 - 3. DCSビューアー/JDCSテストの実行

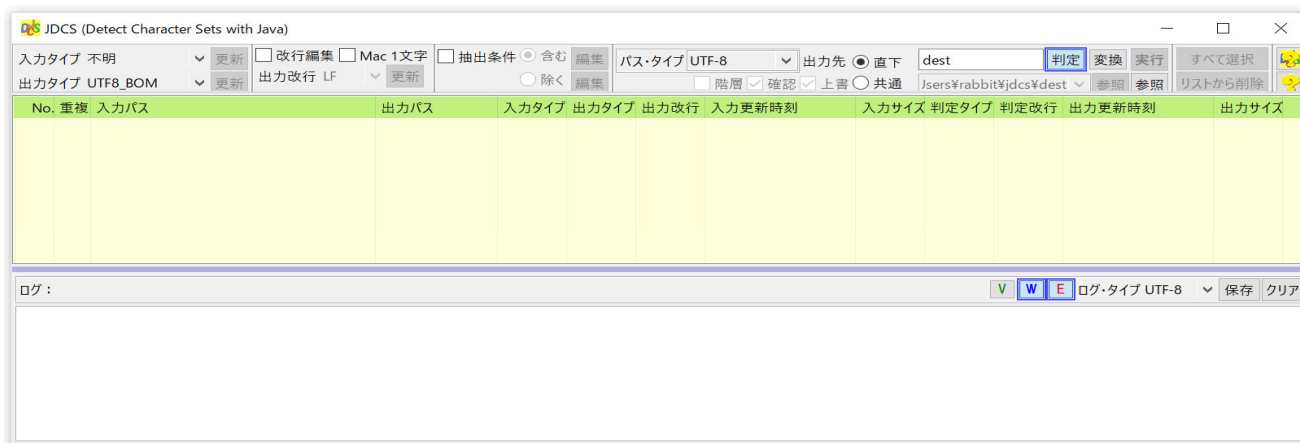
プロジェクト jdc_test では main クラスのある HelloJdcs.java を選択し、メニュー・バー「実行」「実行」「Java アプリケーション」を選択



コンソールにプログラムからのメッセージが表示され、プログラムは終了する

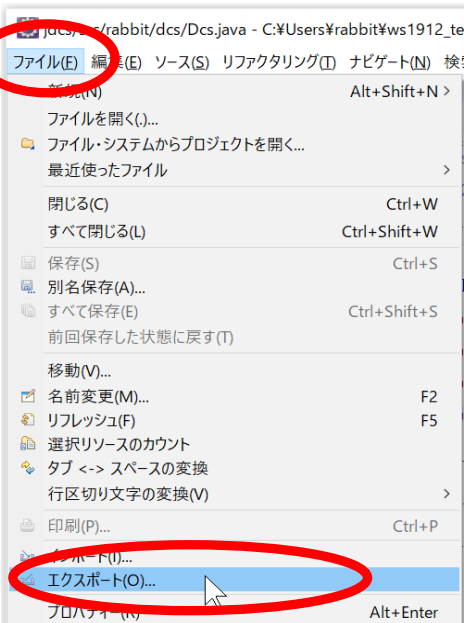


同様にプロジェクト jdcsview では main クラスのある JDCSView.java を選択し、メニュー・バー「実行」「実行」「Java アプリケーション」を選択すると、以下の画面が表示され日本語コード判定／コード変換が行える

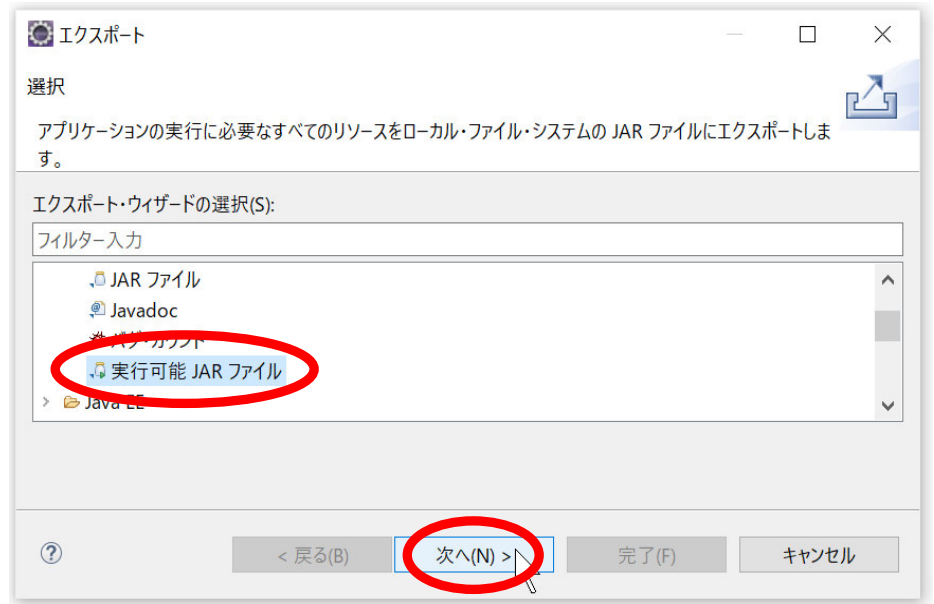


5 - 4. DCSビューアー実行用 JAR 生成／実行

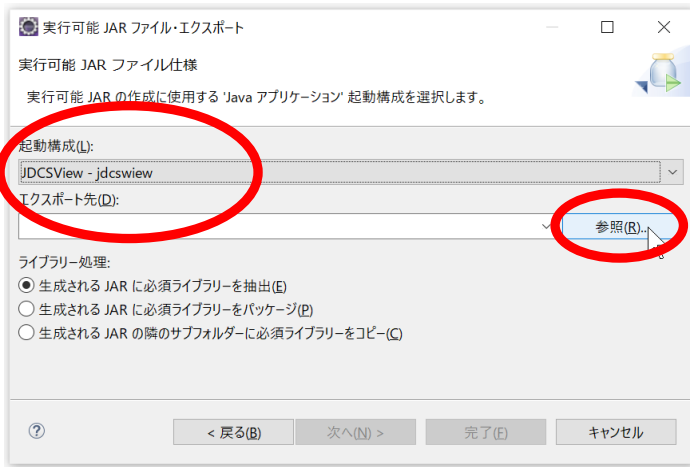
1. 「ファイル」「エクスポート」を選択



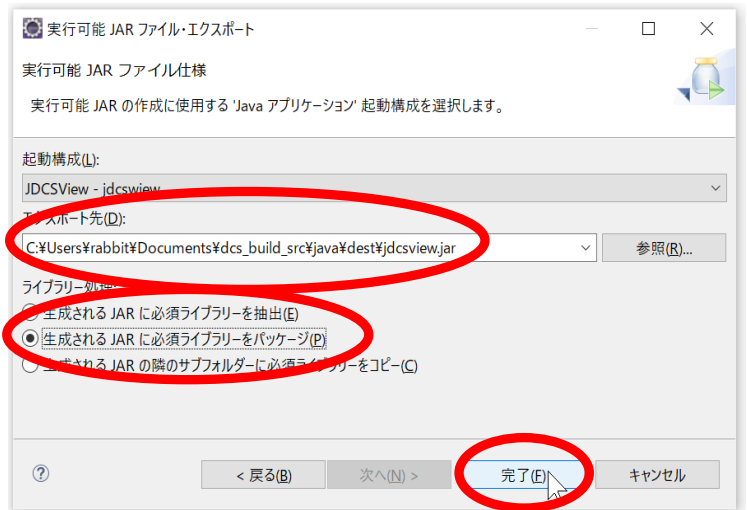
2. ダイアログより「Java」「実行可能 JAR ファイル」を選択して「次へ」



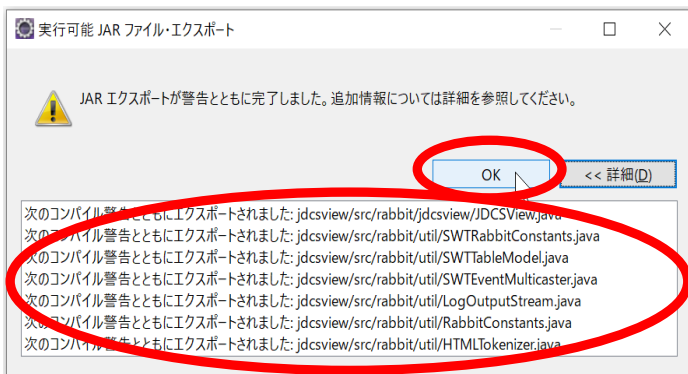
3. 「起動構成」JDCSView を選択し「エクスポート先」「参照」をクリックして、ダイアログから JAR ファイルを設定



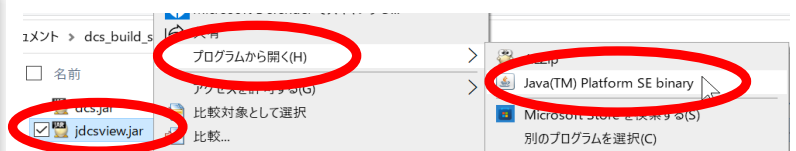
4. エクスポート先のパスを確認して「ライブラリー処理」「生成される JAR に必須ライブラリーをパッケージ」選択し「完了」



5. 以下の 7 ファイルについての警告であれば「OK」



6. jdcswiew.jar をマウス右クリック「プログラムから開く」「Java...」



上記で起動できなければコマンドラインから該当の位置へ移動してから "Java -jar jdcswiew.jar"

```
C:\Users\rabbit\Documents\dc build_src\java\dest>java -jar jdcswiew.jar
```

6. Linux

Linux 系のビルドではWindowsとの差分のみ示します。

6 - 1. DCSプロジェクト

DCS(DLL)は dcs, jdcс を 1 つにまとめた dcs_all というプロジェクトにして、dcs, jdcс フォルダは直接リンクを張る

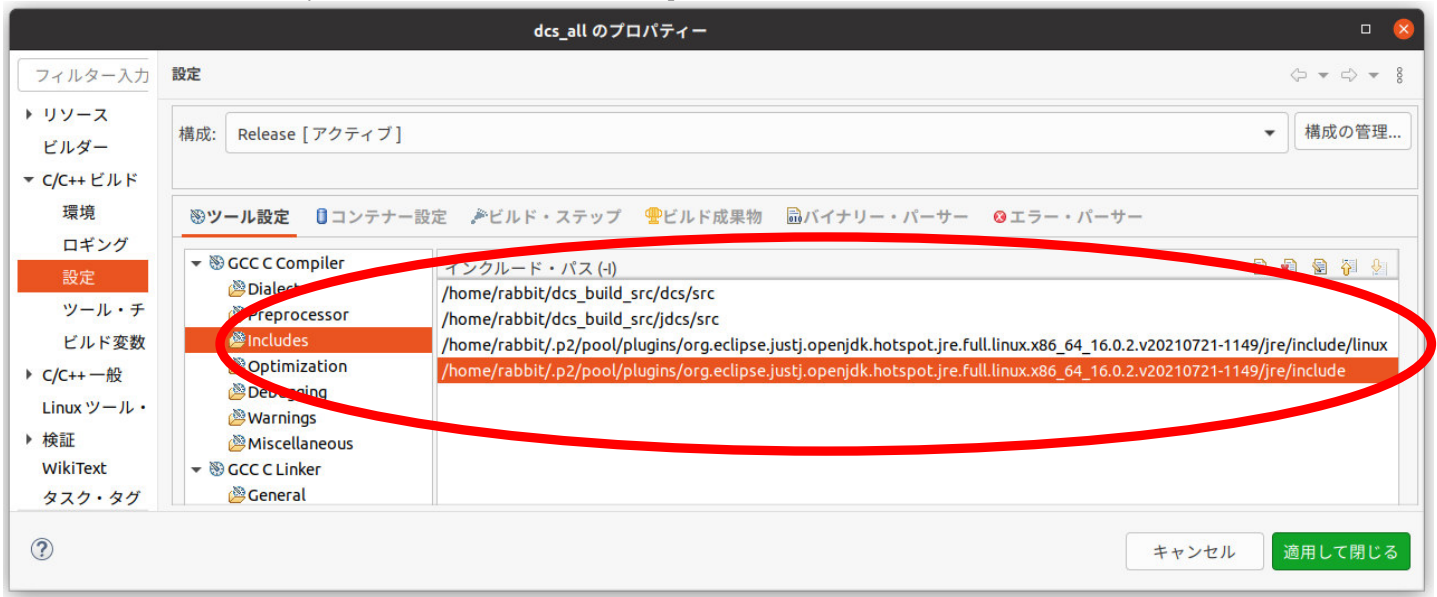
Linuxの共有ライブラリーの「ビルド成果物」では命名規則に合った「成果物の拡張子」(so)、「出力接頭部」(lib) の設定を行なう



Linuxの共有ライブラリーのオブジェクト生成では「位置に依存しないコード(-fPIC)」を設定



DCS(DLL)のJNI インクルード・パスは ~/.p2 配下のものを設定



DCS(DLL)のコンパイラー・オプションは以下の通り

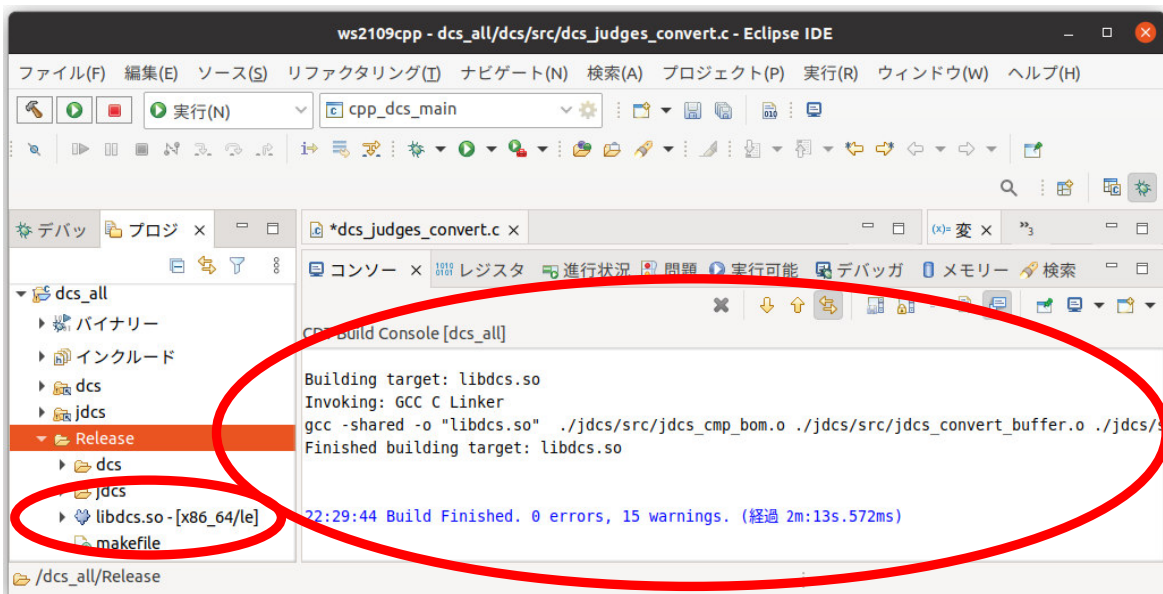
コマンド:	gcc
すべてのオプション:	-I/home/rabbit/dcs_build_src/dcs/src -I/home/rabbit/dcs_build_src/jdcs/src -I/home/rabbit/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_16.0.2.v20210721-1149/jre/include/linux -I/home/rabbit/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_16.0.2.v20210721-1149/jre/include -O3 -Wall -c -fmessage-length=0 -fPIC

要約：
 -I/home/rabbit/dcs_build_src/dcs/src -I/home/rabbit/dcs_build_src/jdcs/src
 -I<JRE のパス>/jre/include/linux -I<JRE のパス>/jre/include
 -O3 -Wall -c -fmessage-length=0 -fPIC

DCS(DLL)のリンカー・オプションは以下の通り

コマンド:	gcc
すべてのオプション:	-shared

DCS(共有ライブラリー)のビルドが正常に終了すると「Release」に libdcs.so が生成される

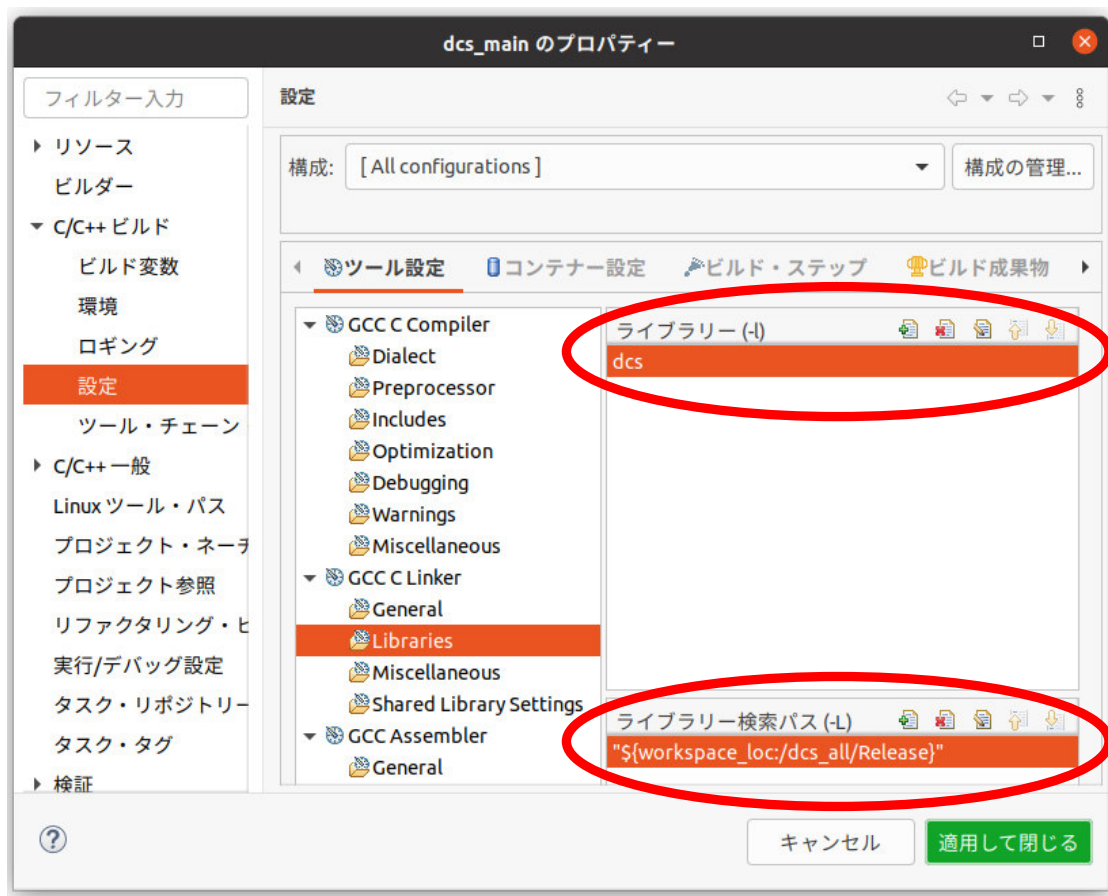


6-2. DCSメイン・プロジェクト

DCSメインでのインクルード・パスは dcs, jdcsc の srcを設定する



DCSメイン・ビルドでのライブラリー検索パスは dcs_all/Release を設定する



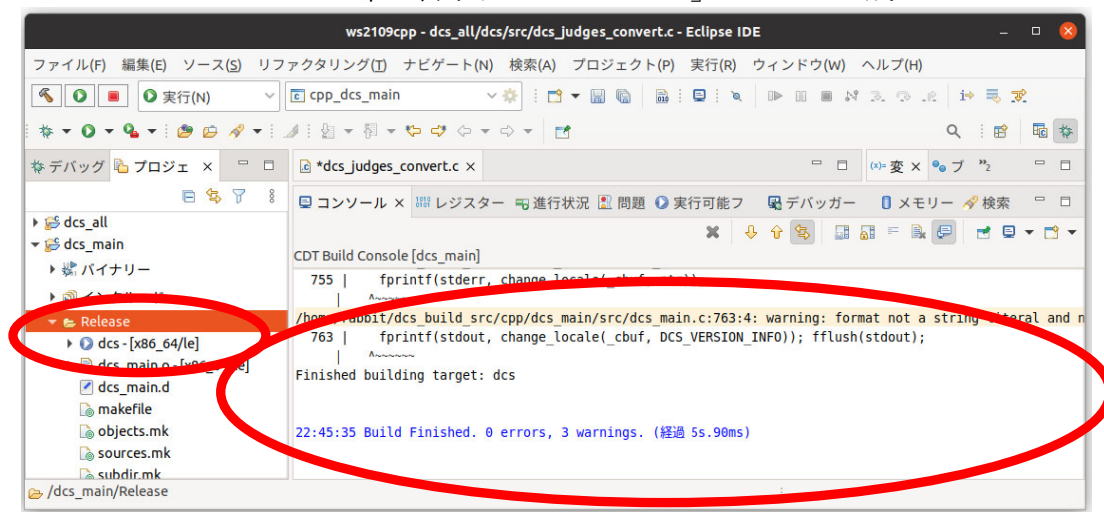
DCSメインのコンパイラー・オプションは以下の通り

コマンド:	gcc
すべてのオプション:	-I/home/rabbit/dcs_build_src/cpp/dcs/src-I/home/rabbit/dcs_build_src/cpp/jdcs/src-O3-Wall-c-fmessage-length=0

DCSメインのリンカー・オプションは以下の通り

コマンド:	gcc
すべてのオプション:	-L"/home/rabbit/ws2109cpp/dcs_all/Release"

DCSメインのビルドが正常に終了すると「Release」に dcs が生成される



6 - 3. DCSビューアー・プロジェクト

jdcsview ビルドでの SWT, JFace のバージョンは以下の通り



7. Android—Termux, UserLAnd(Ubuntu)

7-1. リソース dcs_build_src のSDカードへのコピー

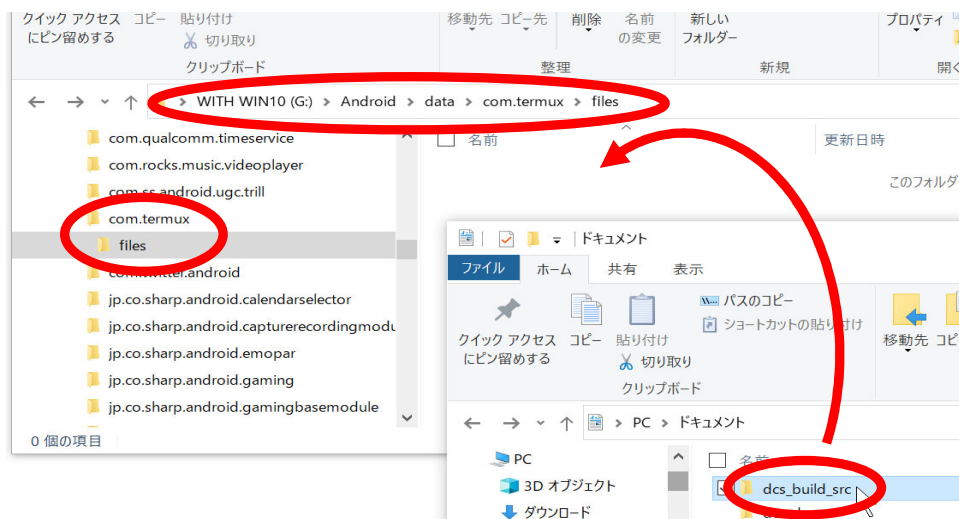
アンドロイド系ではSDカード上でのビルドを前提とします。はじめにPCから以下のパスにリソース・フォルダーdcs_build_srcをコピーした後にTermuxやUserLAnd上のUbuntuでビルドを行ないます。このdcs_build_srcはビルドが完了したら削除して構いません。以後、UserLAnd上のUbuntuを単にUserLAndとします。

PC上のSDカード・パス (Windows Gドライブの場合)

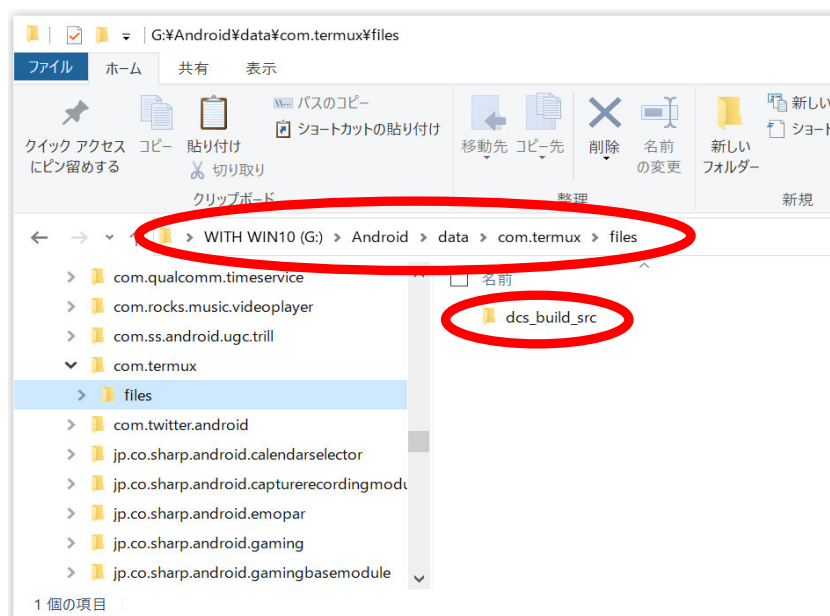
Termux: G:¥Android¥data¥com.termux¥files¥dcs_build_src

UserLAnd: G:¥Android¥data¥tech.ula¥files¥storage¥dcs_build_src

PC上のSDカードにリソース dcs_build_src をドラッグ&ドロップ (Termux がターゲットの場合)



PC上のSDカードにコピーしたリソース dcs_build_src (Termux がターゲットの場合)



7 - 2. Androidでのビルド

リソース dcs_build_src のパスを以下のように想定します。

Termux: ~/storage/external-1/dcs_build_src

UserLAnd: /storage/sdcard/dcs_build_src

実行するシェル・スクリプトはTermux、UserLAndとも以下に示す mklib.sh, mkjar.sh の2本ずつです。これらは別々の環境なのでどちらから実行しても構いません。生成場所は実行フォルダー termux, userland配下であり、他は参照するだけです。

dcs_build_src/	
cpp/	—DCS共有ライブラリー・ソース
java/	—Java JNI ソース
android/	
termux/	
mklib.sh	—dcs, libdcs.so 生成
mkjar.sh	—dcs.jar, hellodcs.jar 生成
userland/	
mklib.sh	—dcs, libdcs.so 生成
mkjar.sh	—dcs.jar, hellodcs.jar 生成
common/	—Termux, UserLAnd 共通リソース

インクルード・ファイル

mklib.sh での共有ライブラリーの生成時に JNI のインクルード・ファイルを参照します。Termux は標準の /usr/include に含まれますが UserLAnd では Java のバージョン名で分かれており、Java が他のバージョンであれば、以下に示す mklib.sh 先頭2行を修正する必要があります。

```
jpath=/usr/lib/jvm/java-17-openjdk-arm64/include
jpath2=/usr/lib/jvm/java-17-openjdk-arm64/include/linux
```

OS名定義

mkjar.sh によるJNI用のクラス生成では、以下のように java/jdcs/rabbit/dcs 配下のソース・ファイル **Dcs.java** にある _os_value の値を_LINUX にする必要があります。mkjar.sh 内で **ed** もしくは **ex** によって編集されますが、両コマンドとも無ければ、どちらかをインストールするか mkjar.sh の実行前に直接編集する必要があります。

修正前：

```
static int _os_value = _WINDOWS;
```

修正後：

```
static int _os_value = _LINUX;
```

7-3. Termux でのビルド

Termux 上でSDカードにアクセスするためには以下のコマンドを1回実行します。

```
$ termux-setup-storage
```

成功すると、以下のようにパス ~/storage/external-1 が参照できるようになります。以下にビルド方法を示していきます。

```
$ cd ~/storage/external-1/dcs_build_src/android/termux           # ビルド場所へ移動
$ bash -x mklib.sh 2>&1 | tee log_mklib.txt                       # dcs, libdcs.so 生成
.
.
.

$ ls -lt
total 3008
-rwxrwx--- 1 u0_a395 everybody 35000 May 15 19:38 dcs           # DCS 実行モジュール
-rwxrwx--- 1 u0_a395 everybody 9974 May 15 19:38 log_mklib.txt
-rwxrwx--- 1 u0_a395 everybody 2901768 May 15 19:38 libdcs.so   # DCS 共有ライブラリー
-rwxrwx--- 1 u0_a395 everybody 922 May 2 09:00 mkjar.sh
-rwxrwx--- 1 u0_a395 everybody 332 May 2 09:00 mklib.sh
$ bash -x mkjar.sh 2>&1 | tee log_mkjar.txt                       # dcs.jar, hellodcs.jar 生成
.
.
.

$ ls -lt . hello jdcs
.:
total 3104
drwxrwx--- 3 u0_a395 everybody 32768 May 15 19:40 hello       # hellodcs.jar (JDcs テストJAR)
-rwxrwx--- 1 u0_a395 everybody 1097 May 15 19:40 log_mkjar.txt
drwxrwx--- 3 u0_a395 everybody 32768 May 15 19:40 jdcs         # dcs.jar
-rwxrwx--- 1 u0_a395 everybody 35000 May 15 19:38 dcs
-rwxrwx--- 1 u0_a395 everybody 9974 May 15 19:38 log_mklib.txt
-rwxrwx--- 1 u0_a395 everybody 2901768 May 15 19:38 libdcs.so
-rwxrwx--- 1 u0_a395 everybody 922 May 2 09:00 mkjar.sh
-rwxrwx--- 1 u0_a395 everybody 332 May 2 09:00 mklib.sh

hello:
total 64
-rwxrwx--- 1 u0_a395 everybody 7069 May 15 19:40 hellodcs.jar  # hellodcs.jar (JDcs テストJAR)
drwxrwx--- 4 u0_a395 everybody 32768 May 15 19:40 rabbit
jdcs:
total 64
-rwxrwx--- 1 u0_a395 everybody 5743 May 15 19:40 dcs.jar       # dcs.jar
drwxrwx--- 3 u0_a395 everybody 32768 May 15 19:40 rabbit
```

```

$ strip libdcs.so dcs                                # シンボル削除
$ cp -p dcs ~/.usr/bin                                # dcs を実行パスにコピー
$ cp -p libdcs.so ~/.lib/libdcs.so.2.1                # libdcs.soをライブラリー・パスにコピー（バージョン付加）
$ cd ~/.lib                                            # ライブラリー・パスに移動
$ ln -s libdcs.so.2.1 libdcs.so                      # 共通名のリンク・ファイル生成
$ ls -l libdcs*                                       # 共通名の確認

lrwxrwxrwx 1 u0_a395 u0_a395      18 May 15 19:45 libdcs.so -> libdcs.so.2.1
-rwxrwx--- 1 u0_a395 everybody 2879832 May 15 19:42 libdcs.so.2.1

$ cd ~/storage/external-1/dcs_build_src/android/termux/hello # hellodcs.jar 実行場所に移動
$ java -jar hellodcs.jar                                     # hellodcs.jar 実行
ms932
[8d a1 93 fa 82 cd 81 41 8a 46 82 b3 82 f1 81 49 ]
今日は、皆さん！
UTF-8
[e4 bb 8a e6 97 a5 e3 81 af e3 80 81 e7 9a 86 e3 81 95 e3 82 93 ef bc 81 ]

$ cp -p hellodcs.jar ../jdc/dcs.jar ~                  # JAR ファイルをホーム・ディレクトリーへ退避

```

7 - 4. UserLand(Ubuntu) でのビルド

mklib.sh などで処理時間の掛かるものはタイムアウトになる恐れがあるので、スクロールなどして時間稼ぎして下さい。

以下の環境変数の設定がなければ ~/.bashrc の末尾に追加してログインし直します。

```

export PATH=usr/local/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH

```

UserLand(Ubuntu)では、パス /storage/sdcard がSDカードとして参照できます。以下にビルド方法を示していきます。

```

$ cd /storage/sdcard/dcs_build_src/android/userland    # ビルド場所に移動
$ bash -x mklib.sh 2>&1 | tee log_mklib.txt             # dcs, libdcs.so 生成
.
.
.

$ ls -lt
total 3072
-rwxrwx---. 1 rabbit 9997    41792  5月 15 20:05 dcs                # DCS 実行モジュール
-rwxrwx---. 1 rabbit 9997    38696  5月 15 20:05 log_mklib.txt
-rwxrwx---. 1 rabbit 9997 2905904  5月 15 20:04 libdcs.so          # DCS 共有ライブラリー
drwxrwx---. 4 rabbit 9997    32768  5月 15 19:52 objs
-rwxrwx---. 1 rabbit 9997     922  5月  2 09:00 mkjar.sh
-rwxrwx---. 1 rabbit 9997     719  5月  2 09:00 mklib.sh

```

```

$ bash -x mkjar.sh 2>&1 | tee log_mkjar.txt                                # dcs.jar, hellodcs.jar 生成
.
.
$ ls -lt . hello jdcs
.:
total 3168
drwxrwx---. 3 rabbit 9997   32768  5月 15 20:07 hello                                # hellodcs.jar (JDCS テストJAR)
-rwxrwx---. 1 rabbit 9997    1097  5月 15 20:07 log_mkjar.txt
drwxrwx---. 3 rabbit 9997   32768  5月 15 20:06 jdcs                                # dcs.jar
-rwxrwx---. 1 rabbit 9997   41792  5月 15 20:05 dcs
-rwxrwx---. 1 rabbit 9997   38696  5月 15 20:05 log_mklib.txt
-rwxrwx---. 1 rabbit 9997 2905904  5月 15 20:04 libdcs.so
drwxrwx---. 4 rabbit 9997   32768  5月 15 19:52 objs
-rwxrwx---. 1 rabbit 9997    922  5月  2 09:00 mkjar.sh
-rwxrwx---. 1 rabbit 9997    719  5月  2 09:00 mklib.sh

hello:
total 64
-rwxrwx---. 1 rabbit 9997  7073  5月 15 20:07 hellodcs.jar                    # hellodcs.jar (JDCS テストJAR)
drwxrwx---. 4 rabbit 9997 32768  5月 15 20:06 rabbit

jdcs:
total 64
-rwxrwx---. 1 rabbit 9997  5746  5月 15 20:06 dcs.jar                        # dcs.jar
drwxrwx---. 3 rabbit 9997 32768  5月 15 20:06 rabbit
$ strip libdcs.so dcs                                                        # シンボル削除
$ cp -p dcs /usr/local/bin                                                  # dcs を実行パスにコピー
$ cp -p libdcs.so /usr/local/lib/libdcs.so.2.1                            # libdcs.soをライブラリー・パスにコピー (バージョン付加)
$ cd /usr/local/lib                                                         # ライブラリー・パスに移動
$ ln -s libdcs.so.2.1 libdcs.so                                            # 共通名のリンク・ファイル生成
$ ls -l libdcs*                                                            # 共通名の確認
lrwxrwxrwx. 1 rabbit rabbit    18  5月 15 20:10 libdcs.so -> libdcs.so.2.1
-rwxrwx---. 1 rabbit  9997 2886864  5月 15 20:09 libdcs.so.2.1
$ cd /storage/sdcard/dcs_build_src/android/userland/hello                # hellodcs.jar 実行場所へ移動
$ java -jar hellodcs.jar                                                  # hellodcs.jar 実行
ms932
[8d a1 93 fa 82 cd 81 41 8a 46 82 b3 82 f1 81 49 ]
今日は、皆さん！
UTF-8
[e4 bb 8a e6 97 a5 e3 81 af e3 80 81 e7 9a 86 e3 81 95 e3 82 93 ef bc 81 ]
$ cp -p hellodcs.jar ../jdcs/dcs.jar ~                                    # JAR ファイルをホーム・ディレクトリへ退避

```


8. Cygwin

8－1. インクルード・パスの設定

ビルド・シェル `mklib.sh` での共有ライブラリーの生成時に JNI 用のインクルード・ファイルを参照していますが、環境に合わせて以下に示す `mklib.sh` 先頭 2 行を修正する必要があります (JRE のバージョンは 1.1 以上であれば可)。

```
jpath=/cygdrive/c/ec2109cpp/eclipse/jre/include
jpath2=/cygdrive/c/ec2109cpp/eclipse/jre/include/win32
```

8－2. DCS のビルド

以下にビルド方法を示します。

```
$ cd /cygdrive/c/Users/rabbit/Documents/dcs_build_src/cygwin          # ビルド場所へ移動
$ bash -x mklib.sh 2>&1 | tee log_mklib.txt                          # dcs.exe, dcs64.dll 生成
    .
    .
    .
$ ls -lt
合計 3157
-rwx--xr-x+ 1 rabbit なし 188234 5月  2 13:47 dcs.exe                # DCS 実行モジュール
-rwx--xr-x+ 1 rabbit なし 3035824 5月  2 13:47 dcs64.dll              # DCS 共有ライブラリー
-rw----r--+ 1 rabbit なし  2242 5月  2 13:47 log_mklib.txt
-rwx-----+ 1 rabbit なし   432 5月  2 13:20 mklib.sh
```