SimutransをWindows Subsystem for Linuxで  
コンパイルしてみる

Windows 10の新機能として追加されたWindows Subsystem for Linux（以後WSL）。

簡単に言えば、WindowsへLinux環境を構築するアプリである。

詳細はMicroSoft公式や各種情報サイトを参考にして頂いて、WSLのセットアップからコンパイル環境構築までを紹介する。

WSLを用いてWindows版Simutransを作成する場合、クロスコンパイルとなる。

なお、記事内で使用しているパソコンのスペック及び導入するディストリビューションは下記の通り。

|  |  |
| --- | --- |
| 機種 | Lenovo ThinkPad X250 |
| OS | Microsoft Windows 10 Pro |
| CPU | Intel Core i5-5200U @ 2.20GHz |
| HDD | 465GB |
| RAM | 8GB |
| WSLディストリビューション | Ubuntu |

なお、特記無き場合下記コマンドのバージョン番号は2018年8月10日時点での最新版のため、実行時には適宜最新版に読み替えて実行して頂きたい。

また、以後のコマンド表記では、

|  |  |
| --- | --- |
| 半角スペース | ␣ |
| 改行 | ⏎ |

で表し、WSL内のユーザ名・PC名は、

|  |  |
| --- | --- |
| ユーザ名 | ren |
| コンピュータ名 | LAPTOP-FJH7CJKU |

で行う。

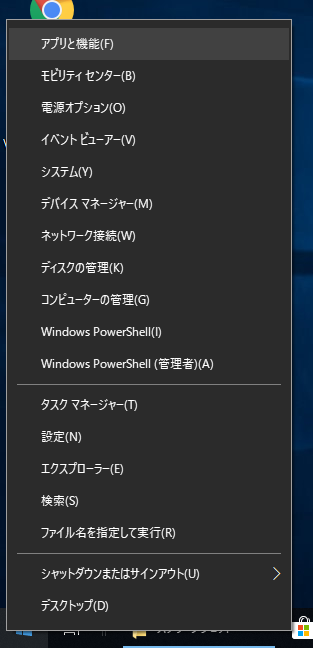
# [1]WSLのセットアップ

## (1)Windows機能のセット

初回利用時は、システム設定を変更するために管理者権限を有するアカウントでログインをし、スタートメニューを右クリックし「アプリと機能(F)」を選択する。

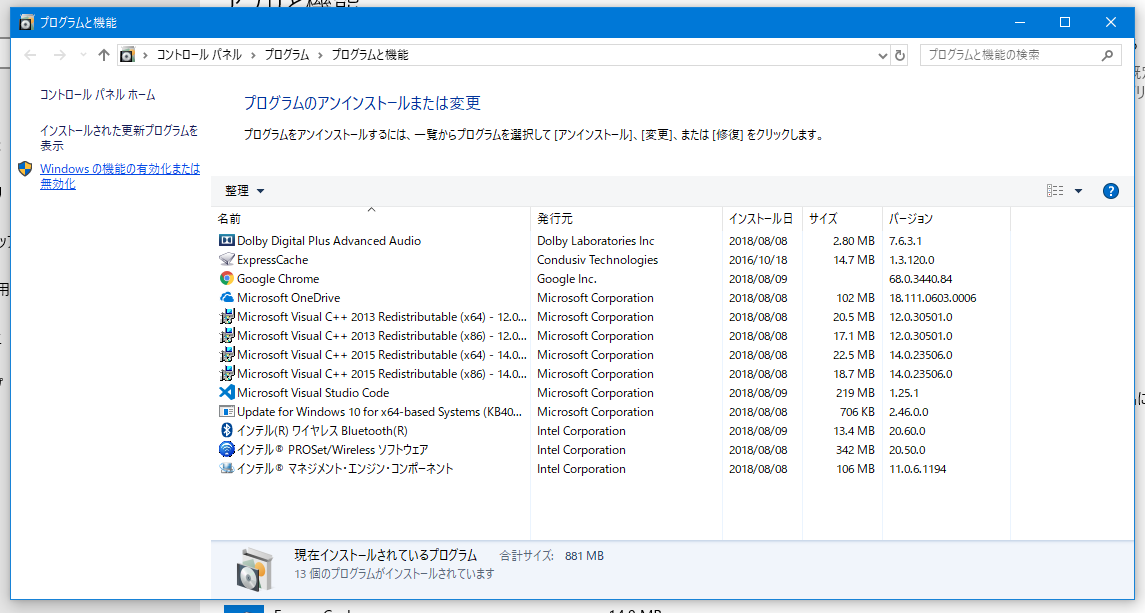
次に、関連設定の「プログラムと機能」をクリックし、コントロールパネルの「プログラムと機能」を呼び出す（図1参照）。

図 1 プログラムと機能の呼び出し



プログラムと機能画面左の「Windowsの機能の有効化または無効化」をクリック（図2赤丸部）し、「Windows Subsystem for Linux」のチェックボックス（図3赤丸部）を  
オンにし、OKをクリックする。

図 2 プログラムと機能画面



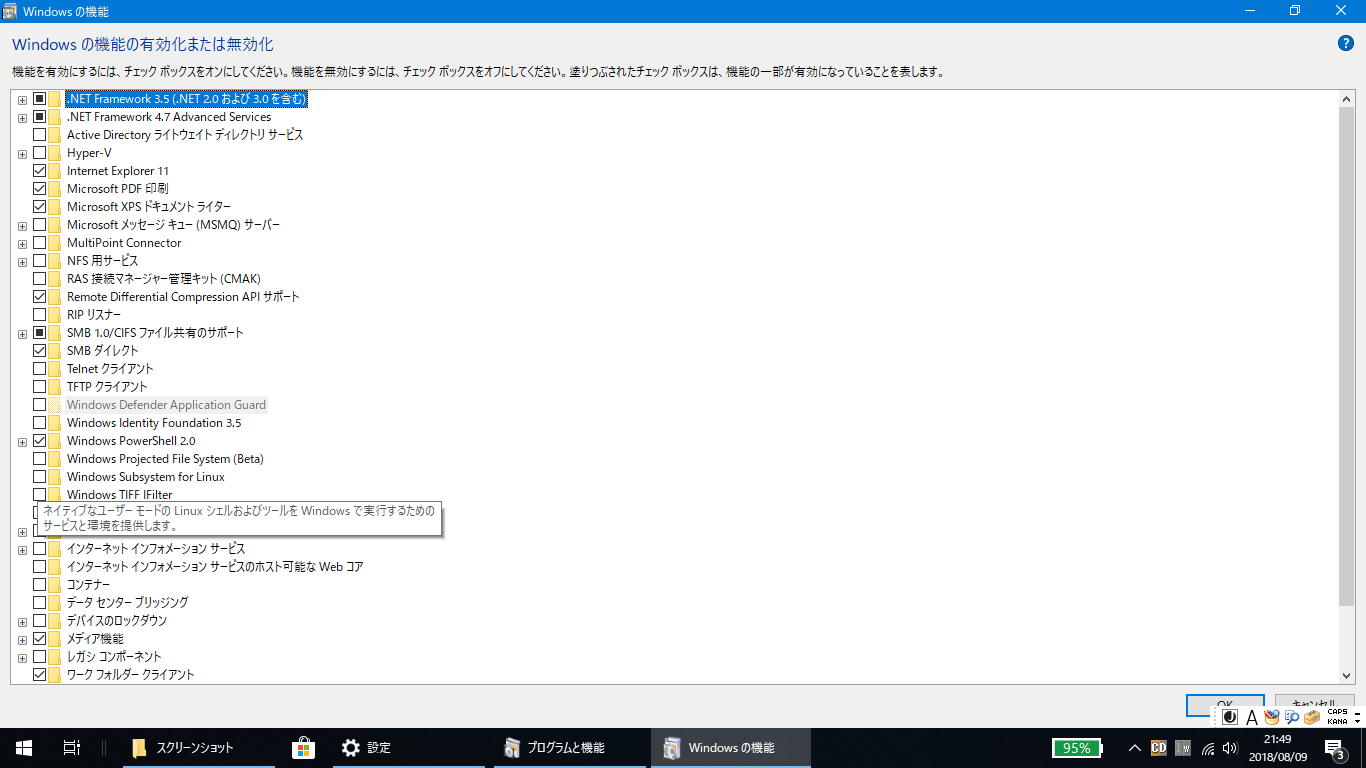


図 3 Windowsの機能画面

設定変更が行われ、再起動を求められるため再起動する（図4赤丸部「今すぐ再起動(N)」をクリック）。

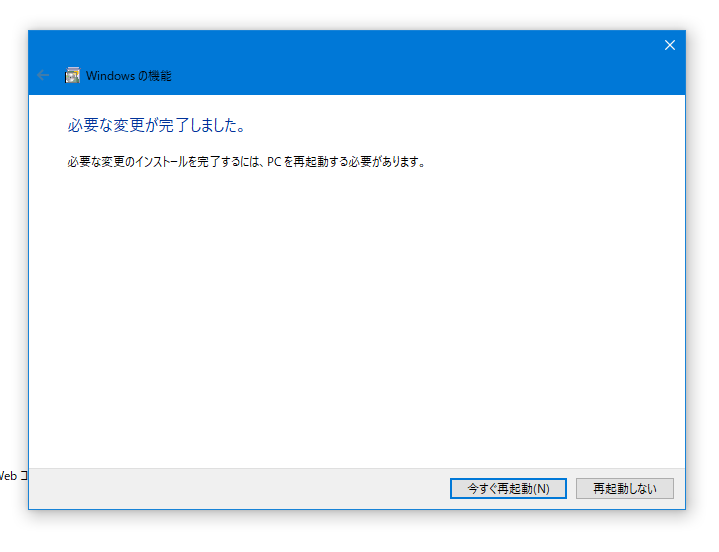
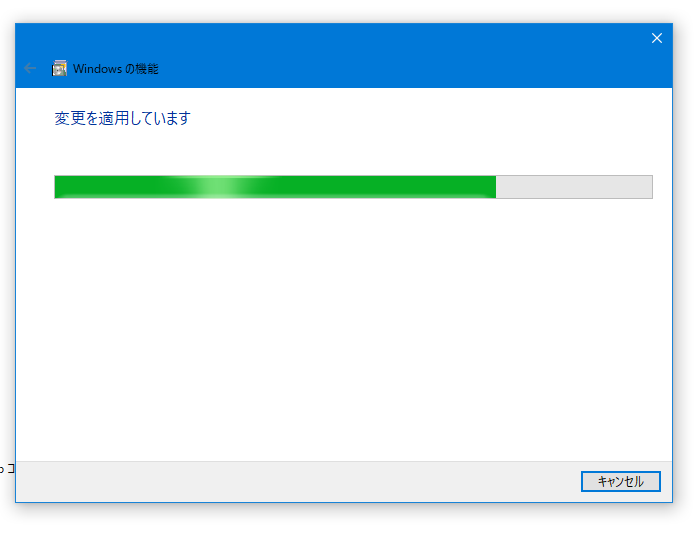


図 4 変更適用中及び適用後画面

## (2)Linuxのインストール

Microsoft Storeを開き、検索窓へ「linux」と入力する（検索窓は「検索」を押すと現れる）。  
現れるリストの中より、「Windows で Linux を実行する」を押す（図5）。

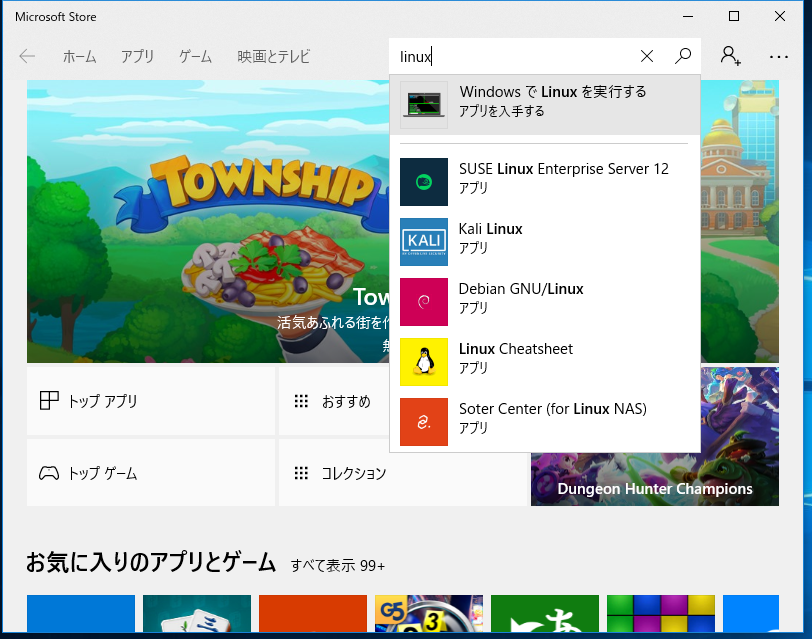
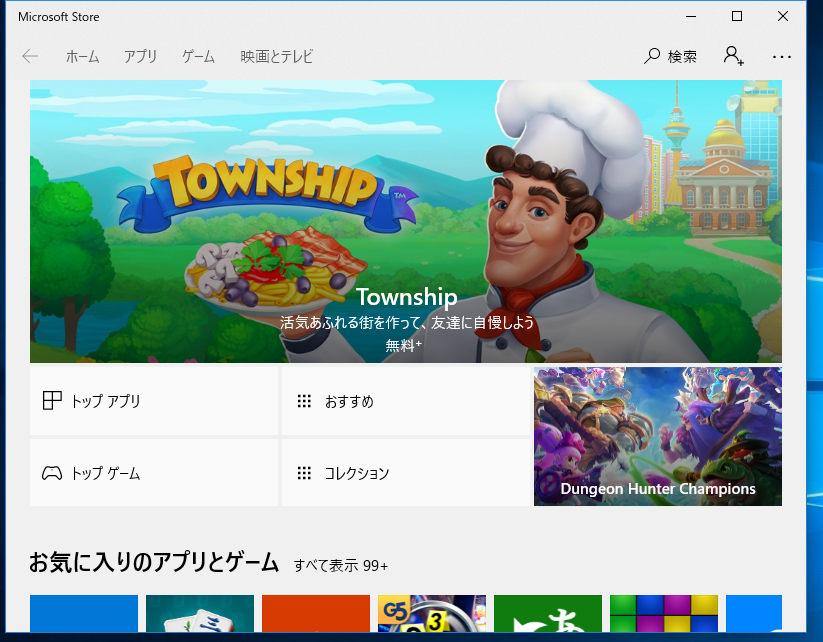


図 5 Microsoft Storeの検索画面

すると、複数のディストリビューションが表示される。今回はUbuntuを導入するため、Ubuntuをクリック（図6赤丸部）し次の画面で「入手」をクリックする（図7）。

図 6 MS SotreのLinux特集画面



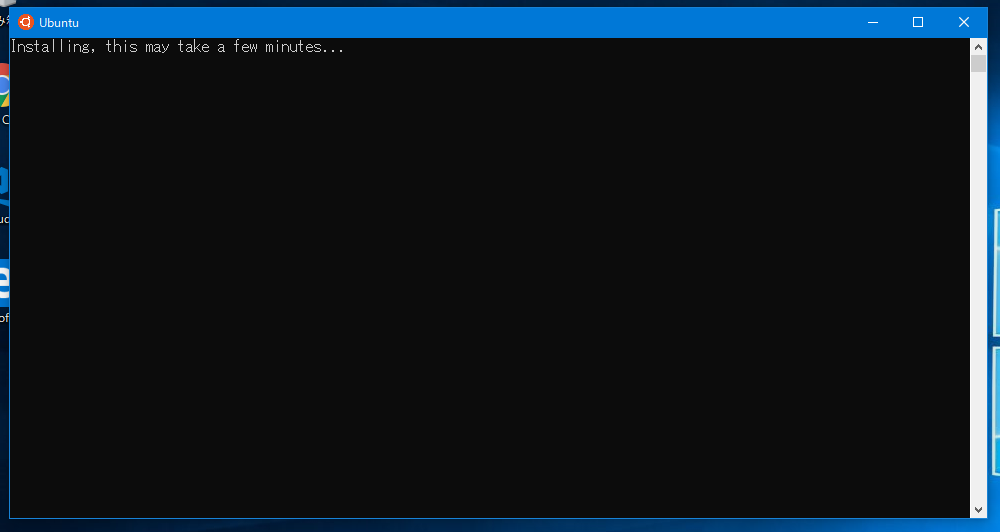
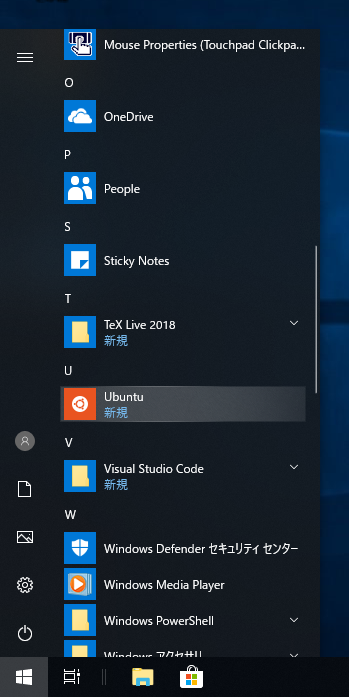
図 7 Ubuntuアプリ詳細画面



インストールが終了したらスタートメニューからUbuntuを探しクリック（またはストア画面から起動を押す）する（図8左赤丸部）。

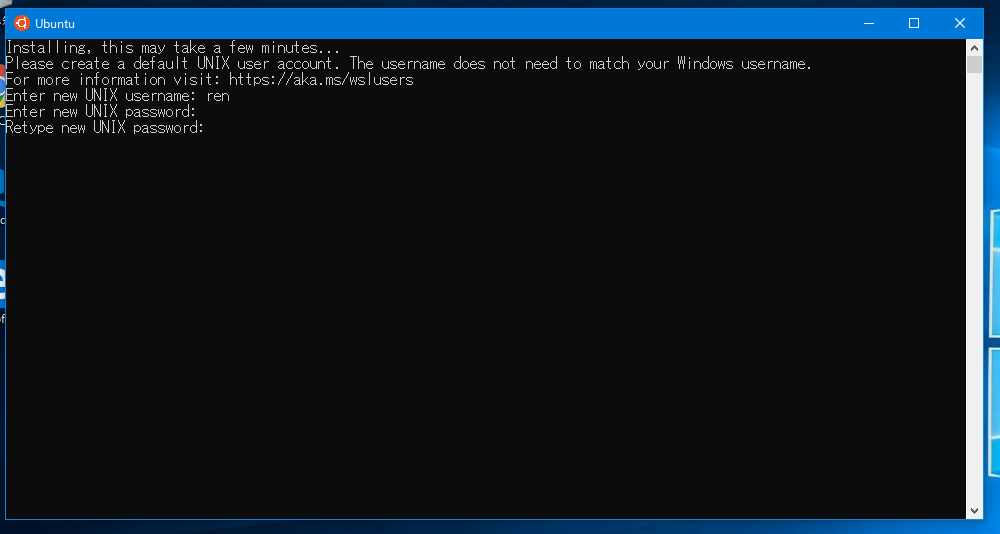
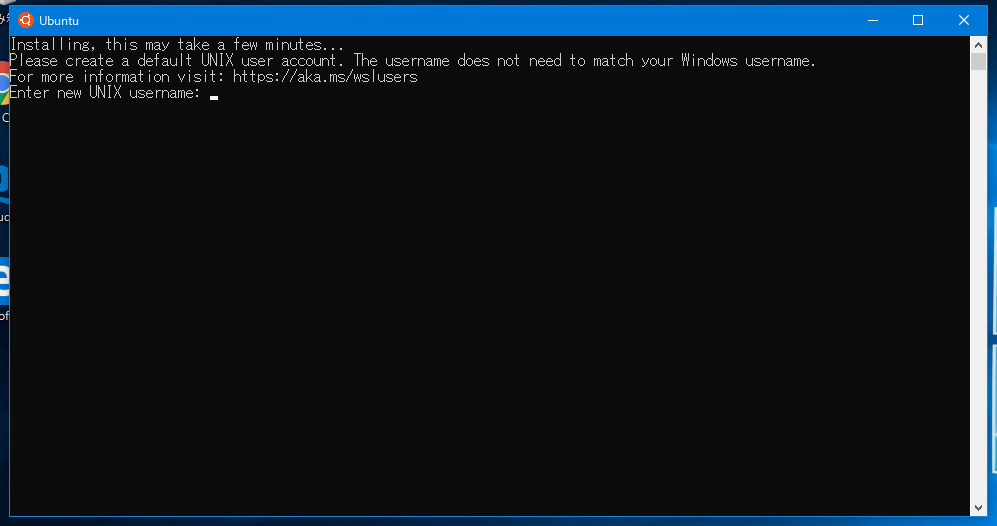
初回起動時は、「Installing, this…」と表示され、ファイルの展開が行われるためしばらく待つ（図8右側）。

図 8 スタートメニューとUbuntuの初回起動画面



ファイルの展開が終了するとユーザ名を聞かれるため、Ubuntu用のユーザ名を入力する。このときWindowsとそろえる必要は無いが英数字で登録する。  
ユーザ名を入力しEnterを押すとパスワードを聞かれる。パスワードは入力しても何も表示されない為注意。パスワードは2回入力して登録完了（図9）。

図 9 左：ユーザ名入力 / 右：パスワード2度目入力



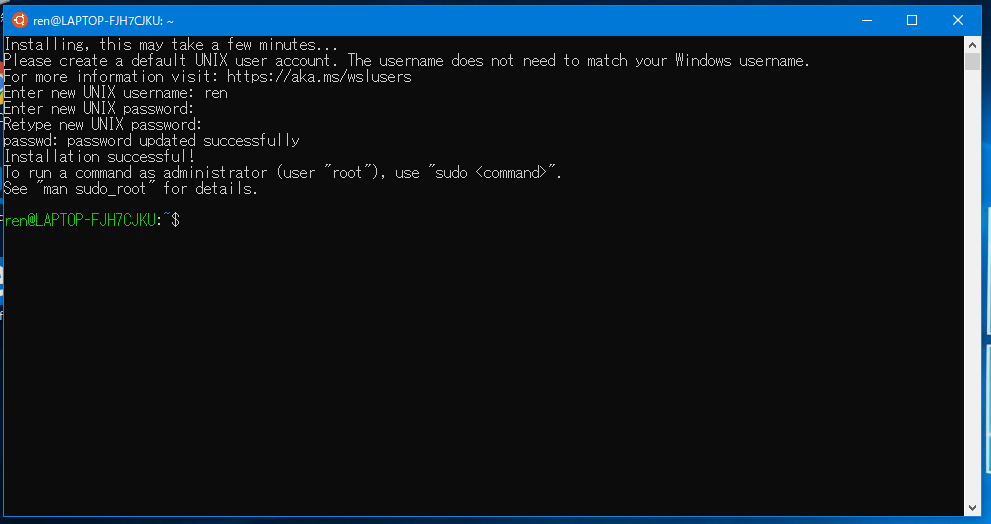
「Enter new UNIX password:」で1回目のパスワード入力、「Retype new UNIX password:」で2回目のパスワード入力を行う。誤った場合は再入力を求められる。

図 10 パスワード設定完了後初期画面

上記のように、:~$ が表示されればWSLセットアップ完了である。

~はユーザのホームディレクトリ（Windows10でいえば C:\Users\(ユーザ名)に相当）を表し、ディレクトリを移動するとこの部分が変わる。最上位は/で表される。WSLを動かしているWindowsのディレクトリは、/mnt/c以下にリンクされている（図11）。

図 11 ディレクトリイメージ

導入完了後のWSLは英語環境となる。日本語化する場合は次の手順で行う。

1. パッケージ更新（update > upgrade）
2. 日本語版パッケージURL・公開鍵の登録>1.を繰り返す
3. 言語パックのインストール
4. ロケールの変更>WSLの再起動
5. タイムゾーン変更
6. マニュアルのインストール

Ubuntu Japaneseチームによる日本版パッケージリストの追加に関しては行わなくてもよく、日本語化も必須では無い。その場合は1.の更新のみを行い、2章へ進む。  
Ubuntuのパッケージ更新はapt updateを必ず先に行い、最新情報を取得後にapt upgradeする（コード1）。ただし、update実行後に下記のようなupgradeメッセージが無い場合は不要なため先へ進めてよい。

|  |
| --- |
| **ren@LAPTOP-FJH7CJKU:~$ sudo␣apt␣update⏎**  **[sudo] password for ren:**  **・・・（中略）・・・**  **\*\* packages can be upgraded. Run 'apt list --upgradable' to see them.**  **ren@LAPTOP-FJH7CJKU:~$ sudo␣apt␣upgrade⏎**  **・・・（中略）・・・**  **After this operation, \*\*.\* kB of additional disk space will be used.**  **Do you want to continue? [Y/n]** |

コード 1 アップデートのコマンドイメージ

その日初めて、またはsudoコマンドを使用後一定時間経過後であればパスワードを求められるため、パスワードを入力しエンターを押す。なお、パスワード入力しても何も表示されないため注意。

upgradeでは、数など条件により上記のように続行確認「Do you want to continue?」が現れる。続行する際はyを押しキャンセルする際はnを押し、エンターを押す。なお、Yと大文字にする必要は無い。  
日本語化に関しては、当紙では触れないため、各自で調査していただきたい。[[1]](#footnote-1)

# [2]パッケージのセットアップ

クロスコンパイル方法自体は複数の方がブログなどに記載しているが、Ubuntuで行うにはパッケージの導入に工夫が必要である。

## (1)aptでの導入

各種記事で用いられているArch Linuxでは開発用パッケージをまとめたbase-develパッケージがあるが、Ubuntuでは現状そのようなセットは無いためbase-devel内の各パッケージを指定して導入する。

Arch Linux公式にパッケージ内容リストがあるためそこを参考にaptを用いて  
導入する。なお、zlibやbzipはmingw用に編集して導入する必要があるためこの段階では導入しない。

また、これとは別にgitなどのパッケージも導入する（表1）。

表 1 導入するパッケージ一覧

|  |  |
| --- | --- |
| **base-devel の内容分（順不同）** | |
| autoconf | automake |
| binutils | bison |
| fakeroot | file |
| findutils | flex |
| gawk | gcc |
| gettext | grep |
| groff | libtool |
| m4 | make |
| pacman | patch |
| pkg-config | sed |
| sudo | texinfo |
| util-linux |  |
| **他必要なパッケージ（順不同）** | |
| apt-get | install |
| nsis | mingw-w64 |
| git |  |

なお、既に導入済な場合はその旨表示後次のパッケージ導入へ移行（最後の場合はapt終了）するため、導入の有無を確認せずとも問題無い。

installコマンドは、下記の様に入力する。sudoを用いるために、パスワードが求められることがある。この場合、ユーザのパスワードを入力しエンターを押す。複数インストールの場合は、情報取得後に消費されるディスク容量提示後に承認を求められるため、yと入力後にエンターを押す。

|  |
| --- |
| **ren@LAPTOP-FJH7CJKU:~$ sudo␣apt␣install␣autoconf␣automake␣binutils␣biscon␣fakeroot␣file␣findutils␣flex␣gawk␣gcc␣gettext␣grep␣groff␣libtool␣m4␣make␣pacman␣patch␣pkg-config␣sed␣sudo␣texinfo␣util-linux␣apt-get␣install␣nsis␣mingw-w64␣git⏎**  **[sudo] renのパスワード:**  **・・・（中略）・・・**  **この操作後に追加で \*\*.\* kB のディスク容量が消費されます。**  **続行しますか? [Y/n]** |

コード 2 開発用パッケージ導入コマンド

## (2)zip系の導入

/usr/local/srcへクロスコンパイル用に設定したzlib、bzip2を導入する。

導入準備のための共通作業として、cdコマンドを用い、ここではsrcディレクトリへ移動する（コード3）。以後このディレクトリで作業する。

|  |
| --- |
| **ren@LAPTOP-FJH7CJKU:~$cd␣/usr/local/src⏎** |

コード 3 作業ディレクトリへの移動

### 1.zlib

zlib公式サイトより最新版をダウンロードする。その後、解凍して解凍先へ移動する。下記コード内の(Ver)にはバージョン番号をそのまま入力（1.2.11なら1.2.11と）する。解凍し、ディレクトリを移動する（コード4）。

|  |
| --- |
| **ren@LAPTOP-FJH7CJKU:/usr/local/src$sudo␣wget␣http://zlib.net/zlib-(Ver).tar.gz⏎**  **ren@LAPTOP-FJH7CJKU:/usr/local/src$sudo␣tar␣xvf␣zlib-(Ver).tar.gz⏎**  **ren@LAPTOP-FJH7CJKU:/usr/local/src$cd␣zlib-(Ver)⏎** |

コード 4 zlib解凍

環境変数をクロスコンパイル用に設定してmakeを実行する（コード5）。

|  |
| --- |
| **ren@LAPTOP-FJH7CJKU:/usr/local/src/zlib-(Ver)$PREFIXDIR=/usr/x86\_64-w64-mingw32⏎**  **ren@LAPTOP-FJH7CJKU:/usr/local/src/zlib-(Ver)$sudo␣make␣-f␣win32/Makefile.gcc␣BINARY\_PATH=$PREFIXDIR/bin␣INCLUDE\_PATH=$PREFIXDIR/include␣LIBRARY\_PATH=$PREFIXDIR/lib␣SHARED\_MODE=1␣PREFIX=x86\_64-w64-mingw32-␣install⏎** |

コード 5 zlibインストール

インストール後、次のbzip2導入前に前のsrcディレクトリへ戻る（コード6）。

|  |
| --- |
| **ren@LAPTOP-FJH7CJKU:/usr/local/src/zlib-(Ver)$cd␣..⏎** |

コード 6 ひとつ上のディレクトリへ移動する

### 2.bzip2

2018年現在、公式サイトは期限切れにより不審サイト化しておりパッケージをダウンロードできない。そのため、Ubuntuのパッケージ検索サイトよりオリジナル版をダウンロードする。先述通り(Ver)にはバージョン番号をそのまま記述する（恐らく1.0.6のまま変化ないと思われるが）。

|  |
| --- |
| **ren@LAPTOP-FJH7CJKU:/usr/local/src$sudo␣wget␣http://archive.ubuntu.com/ubuntu/pool/main/b/bzip2/bzip2\_(Ver).orig.tar.bz2⏎** |

コード 7 bzip2のダウンロード

zlibと同様に解凍し、ディレクトリを移動する。コマンドは先述のzlibと同様のため省略。ただし、解凍後のディレクトリ名が異なり、圧縮ファイルの.orig以降を除いた名前となる。

移動後、幾つかのファイル内容をクロスコンパイル用に編集する。

* bzip2.c：sys\stat.hの逆スラッシュ “ \ ” （フォントにより円マーク ” \ ” になっている可能性もある）をスラッシュ “ / ” へ変える。
* Makefile：変数設定をMingw系に変更する。

下記コード8に書かれているsedコマンド使用する方法でも編集可能だが、不安な場合はCUIテキストエディタ（emacsやviやnano等）での編集も可能である。なお、ディレクトリがユーザのホームディレクトリ等一般ユーザで書き込める場合**以外**は管理者権限を要するためsudoを忘れずにつけること。一般ユーザが書き込めるディレクトリの場合は管理者権限不要である。大体はホームディレクトリとtmpディレクトリ以外はsudoコマンドが必要となる。

sedコマンドで編集する場合は下記の通りである。

|  |
| --- |
| **ren@LAPTOP-FJH7CJKU:/usr/local/src/bzip2\_(Ver)$sudo␣sed␣-i␣'s|sys\\stat.h|sys/stat.h|g'␣bzip2.c⏎**  **ren@LAPTOP-FJH7CJKU:/usr/local/src/bzip2\_(Ver)$sudo␣sed␣-i␣'s|CC=gcc|CC=x86\_64-w64-mingw32-gcc|g'␣Makefile⏎**  **ren@LAPTOP-FJH7CJKU:/usr/local/src/bzip2\_(Ver)$sudo␣sed␣-i␣'s|AR=ar|AR=x86\_64-w64-mingw32-ar|g'␣Makefile⏎**  **ren@LAPTOP-FJH7CJKU:/usr/local/src/bzip2\_(Ver)$sudo␣sed␣-i␣'s|RANLIB=ranlib|RANLIB=x86\_64-w64-mingw32-ranlib|g'␣Makefile⏎**  **ren@LAPTOP-FJH7CJKU:/usr/local/src/bzip2\_(Ver)$sudo␣sed␣-i␣'s|PREFIX=/usr/local|PREFIX=/usr/x86\_64-w64-mingw32|g'␣Makefile⏎** |

コード 8 bzip2ソースコード編集

このあと、makeでlibbz2.aファイルを作成し、/usr/x86\_64-w64-mingw32ディレクトリ下にファイルを配置する（コード9）。

|  |
| --- |
| **ren@LAPTOP-FJH7CJKU:/usr/local/src/bzip2\_(Ver)$sudo␣make␣libbz2.a⏎**  **ren@LAPTOP-FJH7CJKU:/usr/local/src/bzip2\_(Ver)$sudo␣install␣-m␣644␣bzlib.h␣/usr/x86\_64-w64-mingw32/include⏎**  **ren@LAPTOP-FJH7CJKU:/usr/local/src/bzip2\_(Ver)$sudo␣install␣-m␣644␣libbz2.a␣/usr/x86\_64-w64-mingw32/lib⏎** |

コード 9 bzip2インストール

これで、クロスコンパイルの準備は完了となる。

なお、ここまで紹介したaptを用いたパッケージセットアップ及びzip系のセットアップはシェルスクリプトというファイルにまとめてバックアップすることで、WSLをゼロから再セットアップする際にコピーして取り込み、実行権限付与し実行することで入力時間を大幅に短縮することが出来る。

[参照記事]

Tech TIPS：WSLのUbuntu環境を日本語化する(@IT)

http://www.atmarkit.co.jp/ait/articles/1806/28/news043.html

Tech TIPS：Windows 10でLinuxプログラムを利用可能にするWSL（Windows Subsystem for Linux）をインストールする（バージョン1803対応版）(@IT)

http://www.atmarkit.co.jp/ait/articles/1608/08/news039.html

Ubuntuの日本語環境(Ubuntu Japanese Team)

http://www.ubuntulinux.jp/japanese

Simutransをビルドしてみる(Qiita)

https://qiita.com/Aruneko/items/373ed7d135b6f686dbd9

Arch Linux - base-devel(x86\_64) - Group Details

https://www.archlinux.org/groups/x86\_64/base-devel/

絶対領域(AbsoluteArea)の徒然-Windows 64bit 用のffmpegをビルド-(絶対領域氏)

http://absolutearea.blogspot.jp/2012/09/windows-64bit-ffmpeg.html

なんとな～くしあわせ？の日記-Debian Wheezy上でWindows-x64向けバイナリを作成する-(Hiroguki-Nagata氏)

<http://nantonaku-shiawase.hatenablog.com/entry/2013/04/25/003059>

# [3]WSLカスタマイズなど

## nanoの使い方

nanoは、CUI（Character User Interface）で使用できるテキストエディタの  
ひとつである。  
UNIX系・Linux系で有名なemacsやviに比べ、次の特徴が有る。

* 画面上にショートカットが表示されている  
  AltキーはM、Ctrlキーは^で表されているため、終了（^X）であればCtrlキー＋xキーで行える。
* よく使われるショートカットはキーボード上にあるファンクションキー（F1～F12）に設定されており、先述の終了（^X）はF2となる。  
  他のショートカット・ファンクションキー割り当てに関してはヘルプ（F1）で確認できる。
* カーソル移動はメモ帳等と同様に矢印キーで行えるため、操作方法を一から覚える必要が低い。

上記より、CUI初心者でも比較的扱いやすいと思われる。

### ファイルの開き方・閉じ方

ファイルを開く際は、

|  |
| --- |
| **ren@LAPTOP-FJH7CJKU: ~$nano⏎**  **ren@LAPTOP-FJH7CJKU: ~$nano (file名)⏎** |

と2通りある。  
前者のプログラム名のみの場合、保存時にファイル名を決められる。  
後者のファイル名有りの場合、既存のファイル名ならばファイルを開き、新規作成であれば保存時にそのファイル名で保存される。

閉じる（終了）する際は、閉じるコマンド（F2キー又はCtrl+x）を押す。  
保存はF3キー又はCtrl+oで行える。保存後閉じる場合は、F3キー（Ctrl+o）→F2キー（Ctrl+x）で行える。

## シェルスクリプトの例

詳しい書き方は省略させて頂くが、基本的な部分のみ  
・1行目はbashを指定する1文のみ。#!のあとにコマンドを記述する

|  |
| --- |
| **#!/bin/bash⏎** |

・2行目以降にコマンドを記述する  
・保存終了後、実行属性を付ける

ただ保存しただけでは実行できないために行う。  
・属性付与後実行することで、ファイルへ記述したコマンドが実行される。

|  |
| --- |
| **ren@LAPTOP-FJH7CJKU: ~$ chmod␣+x␣(file名)⏎**  **ren@LAPTOP-FJH7CJKU: ~$ ./(file名)⏎** |

なお、管理者権限必要な場合はパスワードを求められる。

1. 必要な場合、Ubuntuバージョンの確認方法は cat /etc/issuieで行える為、控えておけば導入パッケージ判断などで使える。 [↑](#footnote-ref-1)