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の準備からコンパイルまでを紹介する。

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、ユーザ名・PC名は、

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

で行う。

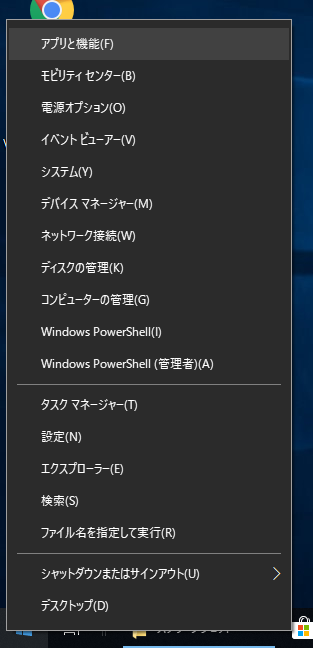
# [1]WSLのセットアップ

## (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。

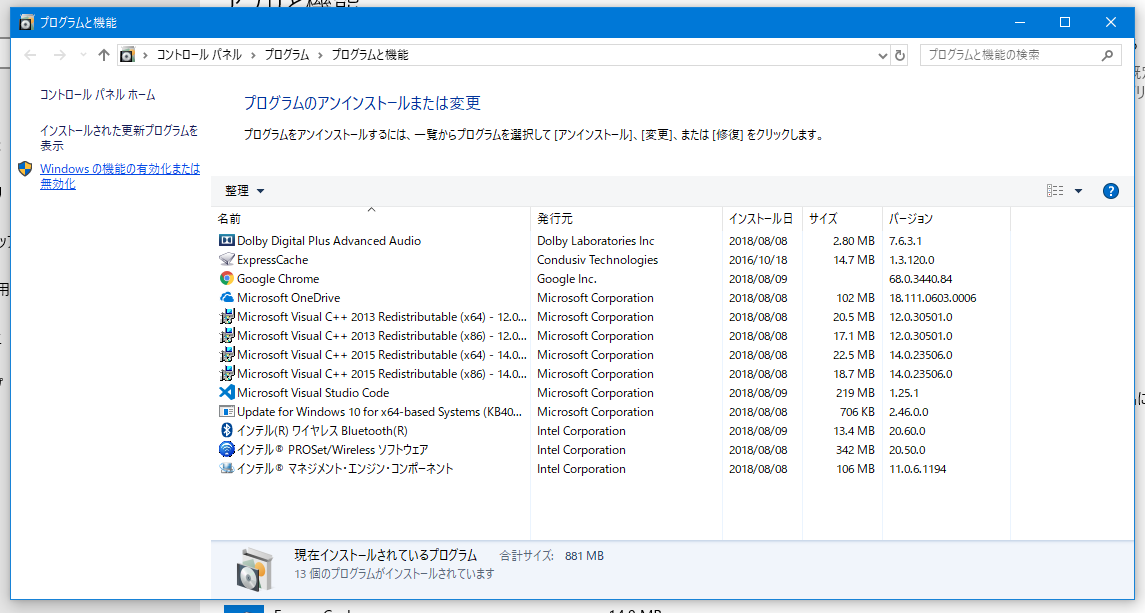
次に、関連設定の「プログラムと機能」をクリックし、コントロールパネルの「プログラムと機能」を呼び出す（図1参照）。

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



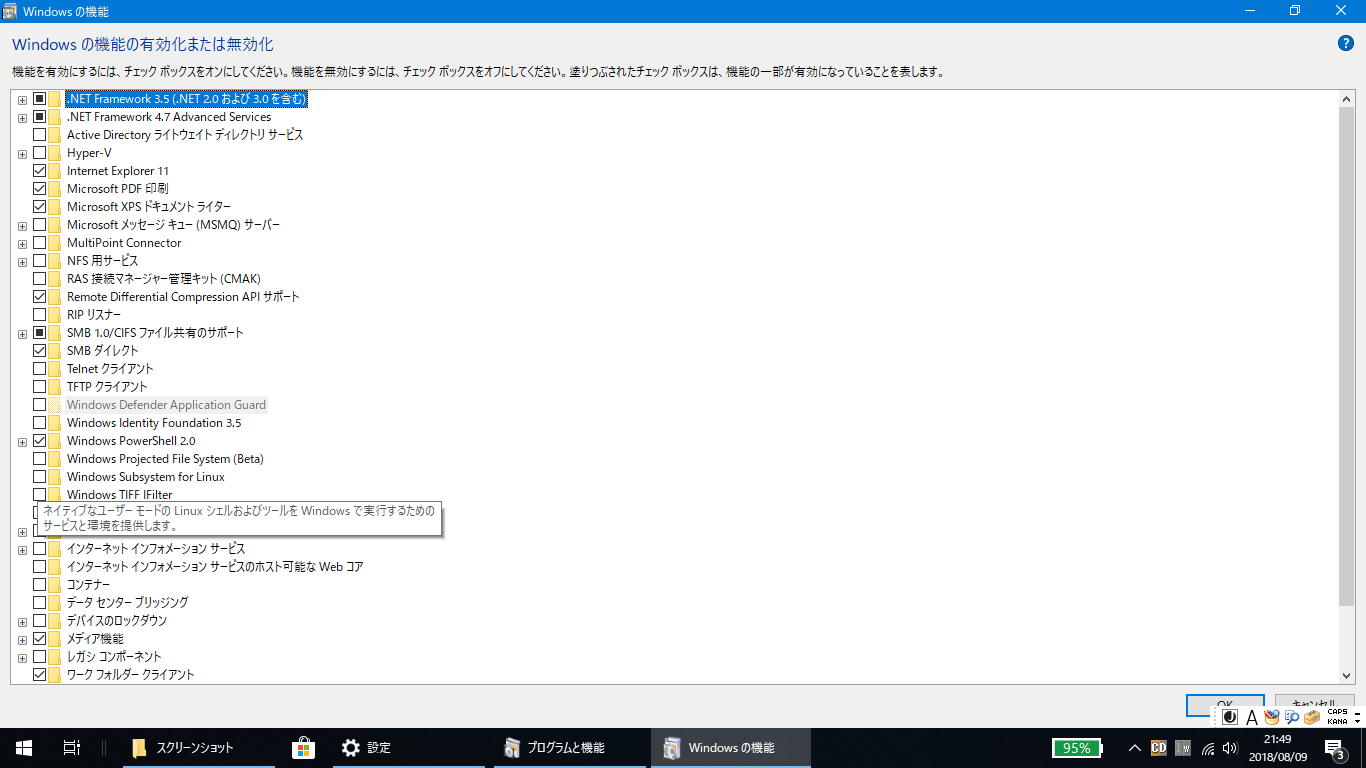


図 3 Windowsの機能画面

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）。

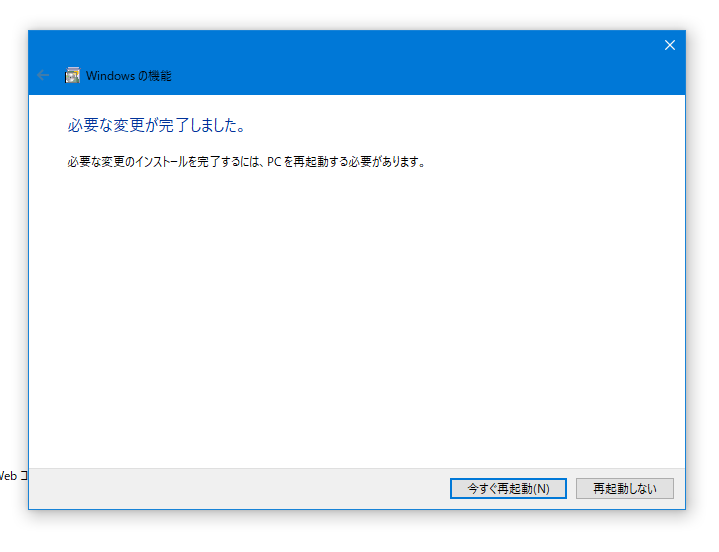
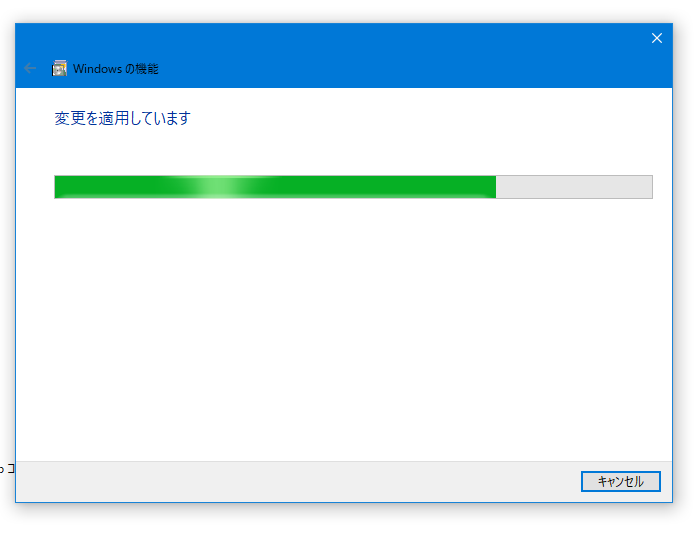


図 4 変更適用中及び適用後画面

## (2)Linuxのインストール

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

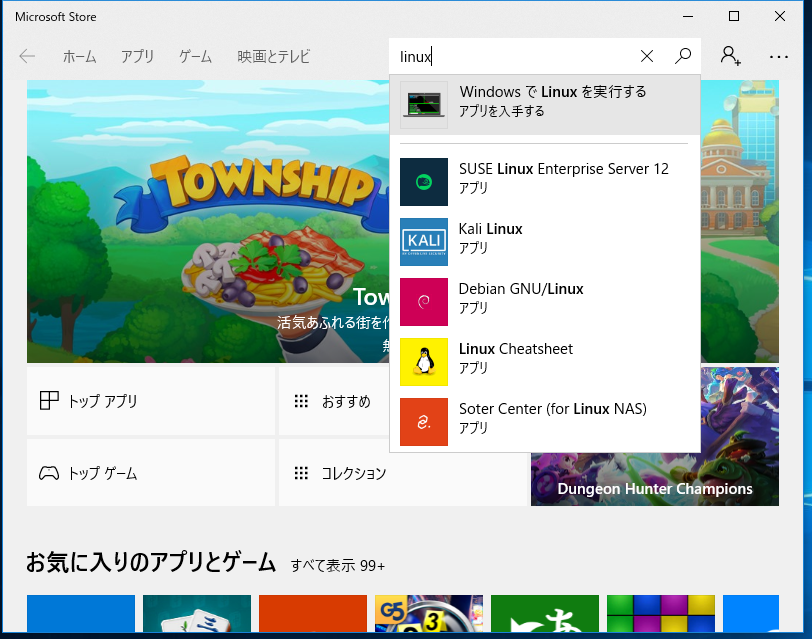
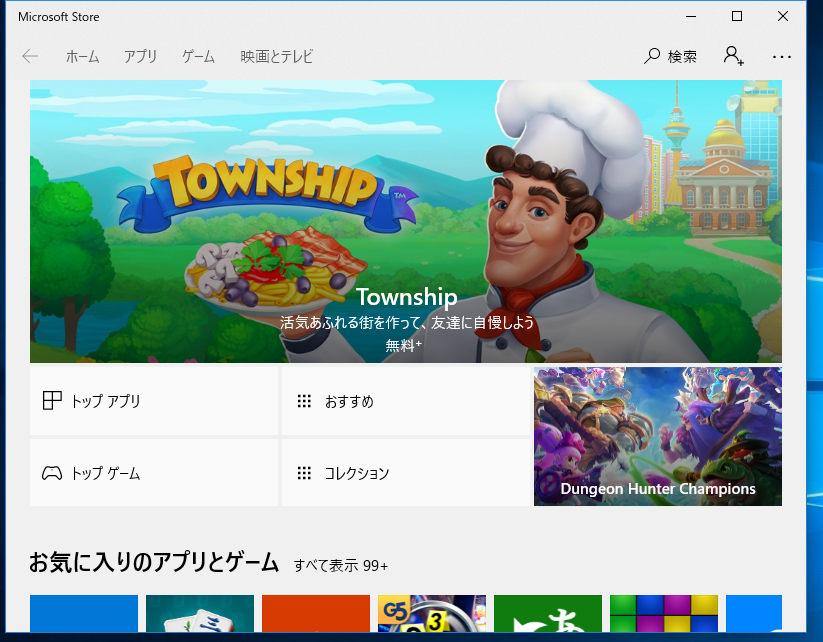


図 5 Microsoft Storeの検索画面

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



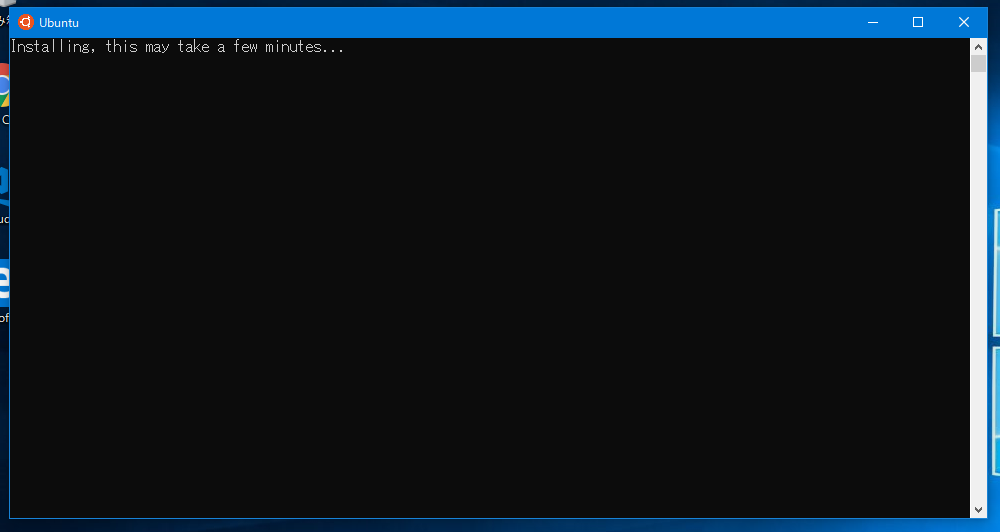
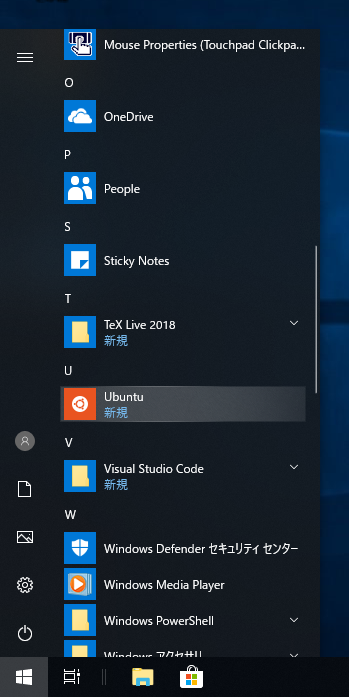
図 7 Ubuntuアプリ詳細画面



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）。

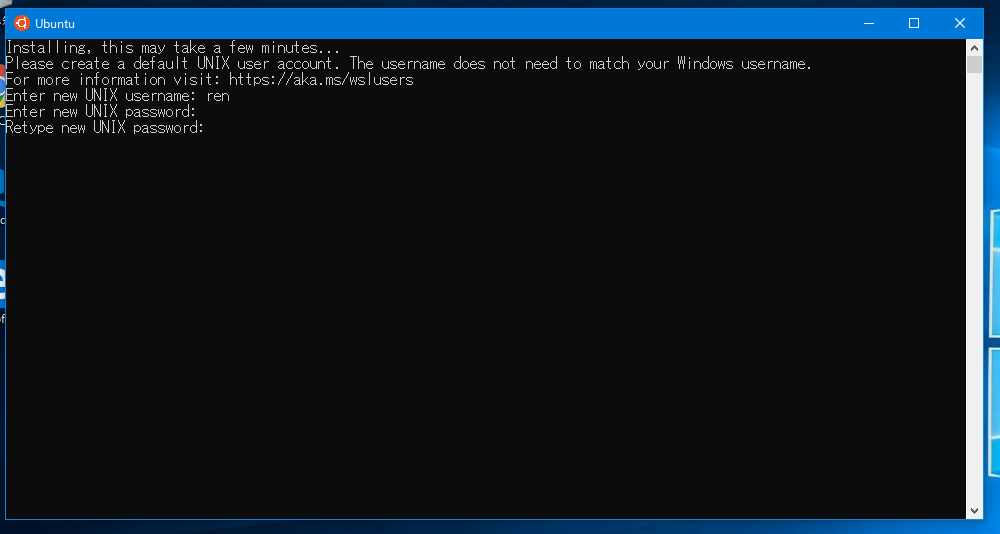
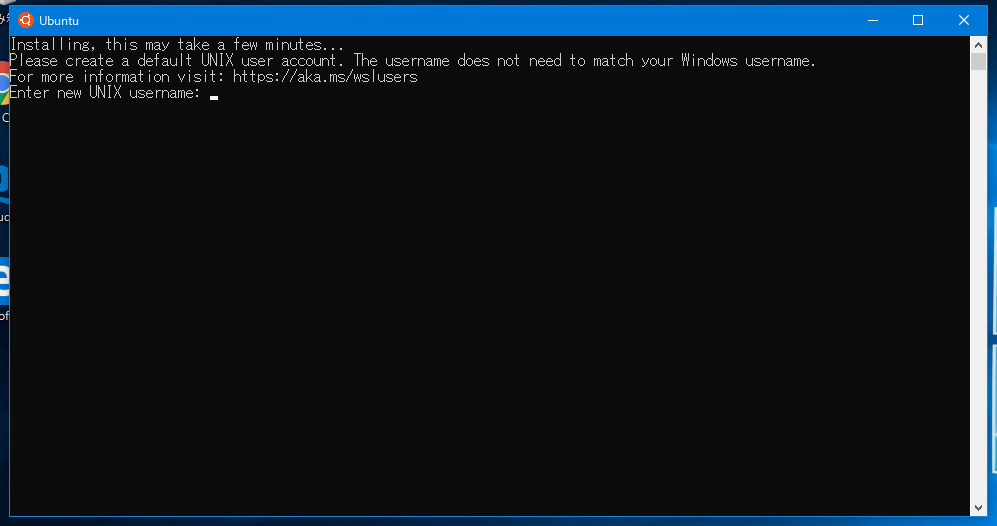
初回起動時は、「Installing, this…」と表示され、ファイルの展開が行われるためしばらく待つ（図8右側）。

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



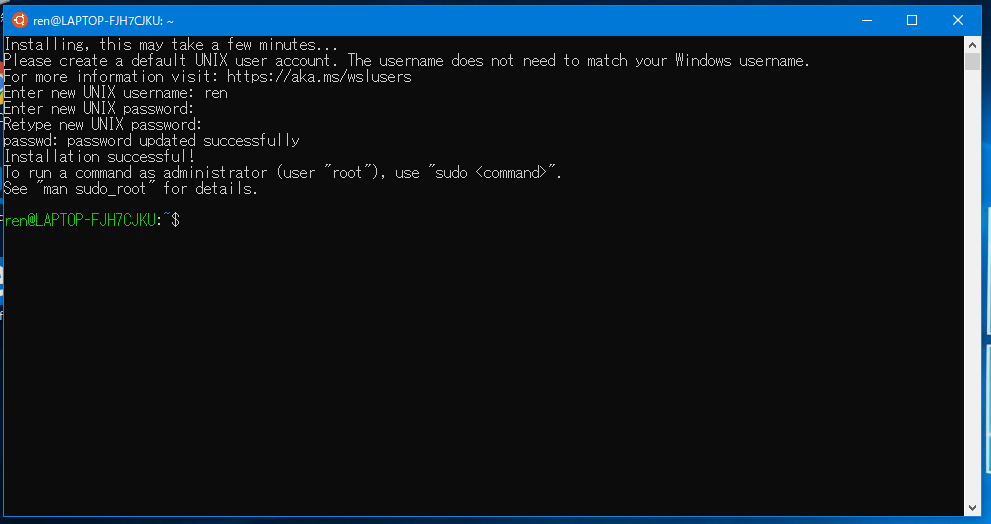
「Enter new UNIX password:」で1回目のパスワード入力、「Retype new UNIX password:」で2回目のパスワード入力を行う。誤った場合は再入力を求められる。

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。Windowsのディレクトリは、/mnt/c以下にリンクされている。

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. (Japanese Teamサイト参考にパッケージリストの追加)
2. パッケージ更新（update > upgrade）
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

1番目のパッケージリストの追加に関しては後で行っても良いし、行わなくてもよい。  
パッケージ更新は必ずapt updateを行い、最新情報を取得後にapt upgradeする。

|  |
| --- |
| **(ユーザ名)@(端末名):~$ sudo␣apt␣update⏎**  **[sudo](ユーザ名)のパスワード:**  **・・・（省略）・・・**  **(ユーザ名)@(端末名):~$ sudo␣apt␣upgrade⏎**  **・・・（中略）・・・**  **この操作後に \*\*.\* kB のディスク容量が解放されます。**  **続行しますか? [Y/n]** |

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。  
詳細は、@ITの記事及びUbuntu Japanese Teamのサイトを参照願いたい。英語環境で問題無い場合は不要であるため、updateとupgradeで問題無い。

Ubuntuバージョンの確認方法は cat /etc/issuieで行える。

## [2]パッケージのセットアップ

Qiitaの記事が詳しいが、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

Qiita記事で用いている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公式にリストがあるためそこを参考にaptで導入する。  
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。

（表を挿入する）

また、これとは別にgitなどのパッケージも導入する。

（コマンドを挿入）

### (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。

共通作業：srcディレクトリへ移動する

|  |
| --- |
| **cd␣/usr/local/src⏎** |

#### 1.zlib

zlib公式サイトより最新版をダウンロードする。

|  |
| --- |
| **sudo wget http://zlib.net/zlib-1.2.11.tar.gz⏎** |

解凍し、ディレクトリを移動する

|  |
| --- |
| **sudo tar xvf zlib-1.2.11.tar.gz⏎**  **cd zlib-1.2.11⏎** |

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。

|  |
| --- |
| **PREFIXDIR=/usr/x86\_64-w64-mingw32⏎**  **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⏎** |

#### 2.bzip

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ダウンロードする。

|  |
| --- |
| **sudo wget http://archive.ubuntu.com/ubuntu/pool/main/b/bzip2/bzip2\_1.0.6.orig.tar.bz2⏎** |

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るため、lsコマンドでディレクトリ一覧を取得し確認後に移動する。

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は、sys\stat.hの\（逆スラッシュ）を/（スラッシュ）へ変える。

Makefileは、変数設定をMingw系に変更する。

これらのコマンドは下記の通りである。

sudo sed -i 's|sys\\stat.h|sys/stat.h|g' bzip2.c

sudo sed -i 's|CC=gcc|CC=x86\_64-w64-mingw32-gcc|g' Makefile

sudo sed -i 's|AR=ar|AR=x86\_64-w64-mingw32-ar|g' Makefile

sudo sed -i 's|RANLIB=ranlib|RANLIB=x86\_64-w64-mingw32-ranlib|g' Makefile

sudo sed -i 's|PREFIX=/usr/local|PREFIX=/usr/x86\_64-w64-mingw32|g' Makefile

[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