

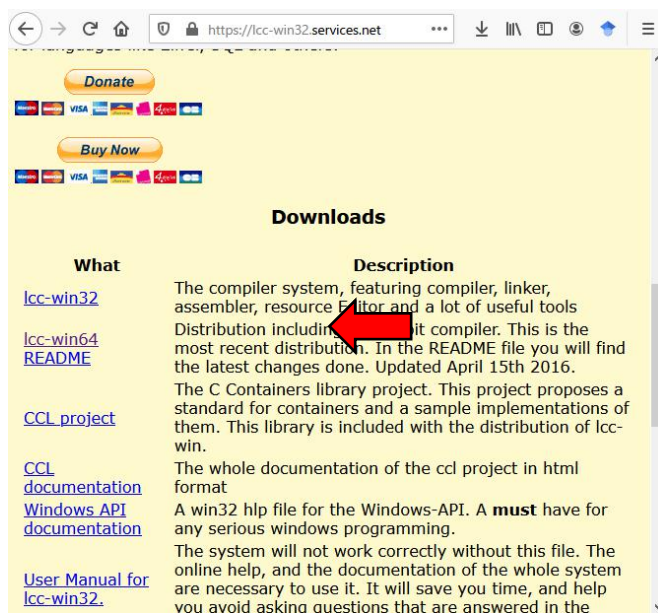
手元にプログラミング処理系などが無い人のための最低限の課題実行環境

## 1. 実行環境

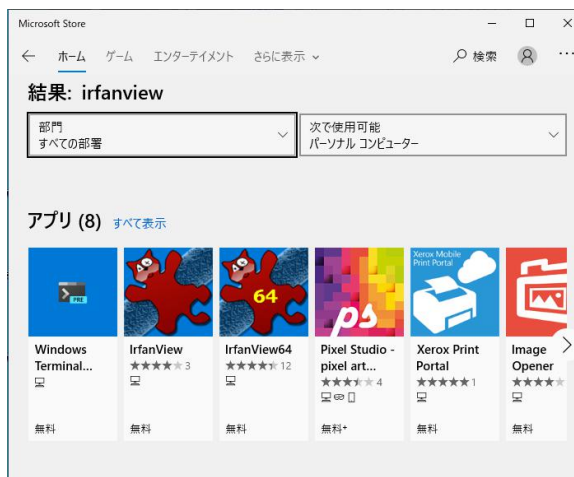
Windows (32 ビットまたは 64 ビット)を想定しています。

## 2. 環境準備

- LCC-Win32 または LCC-Win64 を使います。
- <https://lcc-win32.services.net/> にアクセスして、LCC-Win32 または Win64 のインストーラーをダウンロードします。



- ファイル `lccwin32.exe` または `lccwin64.exe` がダウンロードされてくるので、これを実行するとインストールされます。
- PPM ファイルを見るためのプログラムをインストールします。Microsoft Store で IrFanView を検索して、IrFanView または IrFanView64 をインストールします。



### 3. 必要なファイルのダウンロード

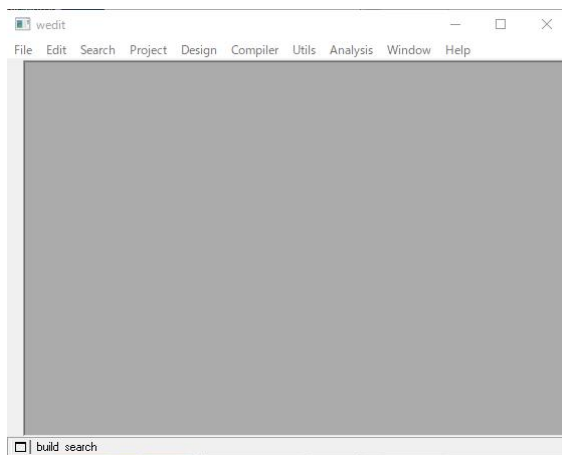
- 課題に使えるプログラム `proc_w.c`, `256-1.ppm` をダウンロードして、適当なフォルダに入れます。

[https://github.com/akinori-ito/wavelet-image-example/proc\\_w.c](https://github.com/akinori-ito/wavelet-image-example/proc_w.c)

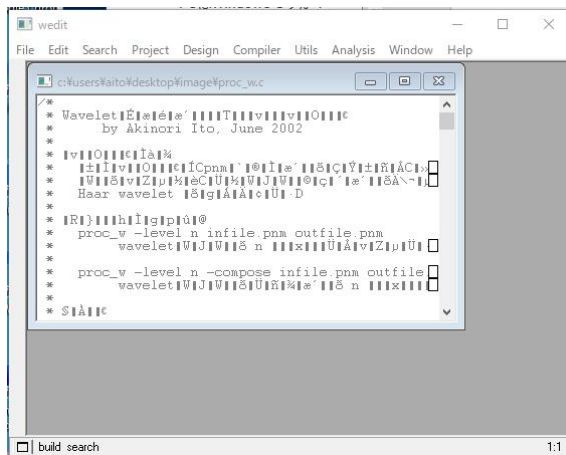
<https://github.com/akinori-ito/wavelet-image-example/256-1.ppm>

### 4. とりあえずコンパイルしてみる

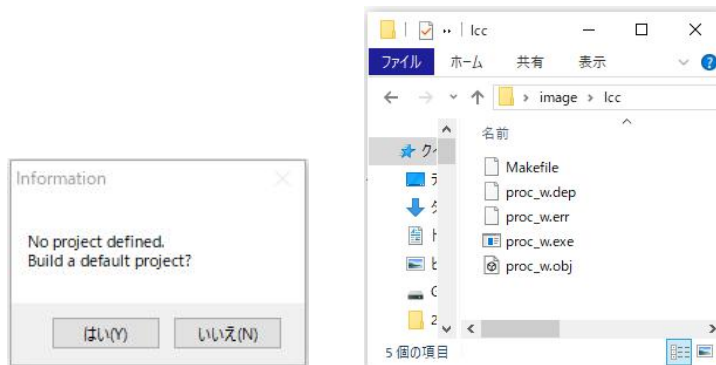
- スタートメニューから `lcc-win > lcc-win32` （または `lcc-win64`）を起動。Wedit というテキストエディタが開く。



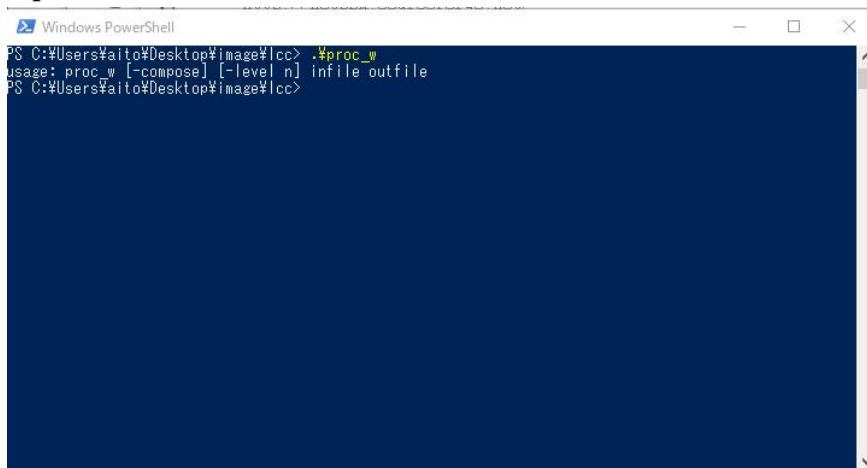
- [File] > [Open...] でファイルダイアログを開き、`proc_w.c` を開く。コメントが文字化けしているが、気にしなくて良い。



- [Compiler] > [Make] を実行。下のようなダイアログが開くので、[はい]を選ぶ。  
Proc\_w.c があるフォルダの下に lcc というフォルダができ、コンパイルした proc\_w.exe ができる。



- エクスプローラで lcc のフォルダを表示し、[ファイル] > [PowerShell を開く] で PowerShell のウィンドウを開く。
- .\proc\_w [enter] で実行してみる。コマンドの使い方が印刷される。

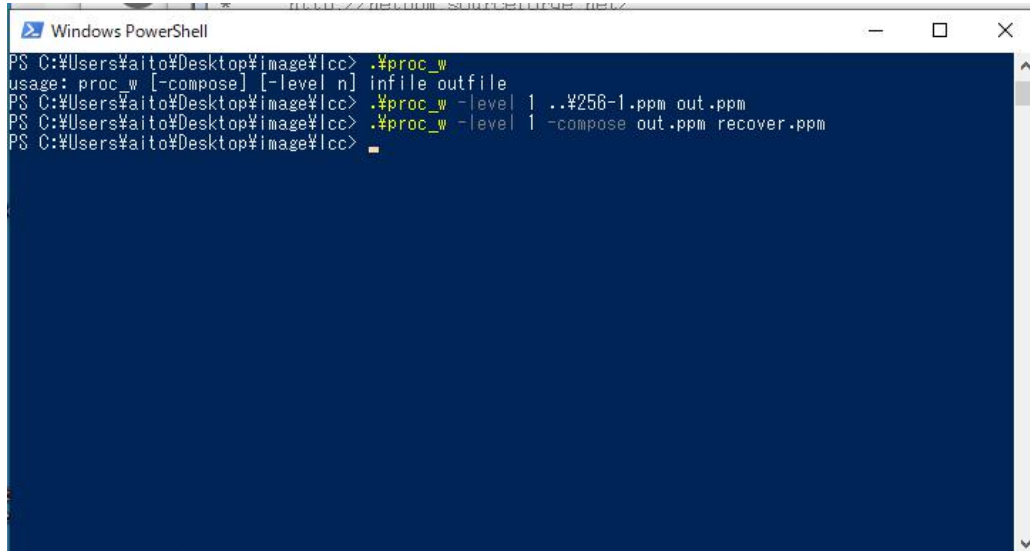


- コマンド proc\_w の使い方
  - .\proc\_w -level 1 infile.ppm outfile.ppm で、infile.ppm を wavelet で分解して outfile.ppm に出力する。-level オプションは、何レベルまで wavelet で分

解するかを指定する。

- `.\proc_w -level 1 -compose infile.ppm outfile.ppm` で、wavelet 分解して得られた `infile.ppm` を元のファイル `outfile.ppm` に戻す。

- 下記のように入力して、`out.ppm` と `recover.ppm` を作ってみよう。



```
Windows PowerShell
PS C:\Users\aito\Desktop\image\lcc> .\proc_w
usage: proc_w [-compose] [-level n] infile outfile
PS C:\Users\aito\Desktop\image\lcc> .\proc_w -level 1 ..\256-1.ppm out.ppm
PS C:\Users\aito\Desktop\image\lcc> .\proc_w -level 1 -compose out.ppm recover.ppm
PS C:\Users\aito\Desktop\image\lcc>
```

## 5. 課題では何をすればいいのか

課題では、単に wavelet 分解して再構成するだけでなく、wavelet 係数の下位何ビットかを 0 にして画質を落とすことをする必要があります。

関数 `main()` のなかに次の部分があります。

```
if (compose) {
    int w,h;
    for (i = level-1; i >= 0; i--) {
        w = width/(1<<i);
        h = height/(1<<i);
        do_compose(ppm,w,h);
    }
} else {
    for (i = 0; i < level; i++) {
        do_wavelet(ppm,width,height);
        width /= 2;
        height /= 2;
    }
}
// ここまで wavelet 係数が ppm に入っている
if (MyPPM_write(ppm,outfile) == NULL) {
    fprintf(stderr,"Can't open %s\n",outfile);
    return 1;
}
```

このなかの `if (MyPPM_write(...))` の直前で、ファイルに書き込む wavelet 係数が ppm の中に入っているので、ここの部分で「ppm の wavelet 係数の下位 n ビットを 0 にする」という処理を入れれば良いです。

変数 ppm の幅は ppm->width, 高さは ppm->height に入っており、座標(x,y)、色 c ( $0 \leq c < 3$ ) は MyPPM\_point(ppm, x,y)[c] でアクセスすることができます。

実際の処理は、例えばこんな感じになります。

```
int cutbit = 3; // 下 3 ビットを 0 にする
int div = 1<<cutbit;
for (int i = 0; i < ppm->width; i++) {
    for (int j = 0; j < ppm->height; j++) {
        for (int c = 0; c < 3; c++) {
            PPM_point(ppm,i,j)[c] /= div;
            PPM_point(ppm,i,j)[c] *= div;
        }
    }
}
```

カットするビット数と、wavelet 分解のレベルを変えて分解を行い、分解した画像を zip 圧縮して大きさを測ります。また、それを元に戻した画像を作り、画質がどうなるかを検討します。