# ソフトウェア設計及び実験
# 第三回レポート

6119019056 山口力也

2019/05/07 日提出

## 1　ライブラリの実用例

ライブラリの実用例を 1 つ調べ, それについて文章でまとめ, 自作のプログラムにどうようにしようしたいか 1000 字程度で作文せよ.

自分は高専時代に機械学習を勉強していたので, そこで用いたライブラリをいくつか挙げたいと思う

- Numpy
- Matpplotlib
- Keras

まずは Numpy について説明する.Numpy は,Python において数値計算を効率的に行うためのライブラリである. 厳密にいうとモジュールに当たる. 効率的な数値計算を行うための型付きの多次元配列（例えばベクトルや行列などを表現できる）のサポートを Python に加えるとともに, それらを操作するための大規模な高水準の数学関数ライブラリを提供している. Python は動的型付け言語であるため, プログラムを柔軟に記述できる一方で純粋に Python のみを使って数値計算を行うと, ほとんどの場合 C 言語や Java などの静的型付き言語で書いたコードに比べて大幅に計算時間がかかる. そこで NumPy は,Python に対して型付きの多次元配列オブジェクト (numpy.ndarray) と, その配列に対する多数の演算関数や操作関数を提供することにより, この問題を解決しようとしている.NumPy の内部は C 言語 (および Fortran) によって実装されているため非常に高速に動作する. したがって, 目的の処理を, 大きな多次元配列（ベクトル・行列など）に対する演算として記述できれば（ベクトル化できれば）, 計算時間の大半は Python ではなく C 言語によるネイティブコードで実行されるようになり大幅に高速化する. さらに、NumPy は BLAS API を実装した行列演算ライブラリ (OpenBLAS, ATLAS, Intel Math Kernel Library など) を使用して線形代数演算を行うため,C 言語で単純に書

いた線形代数演算よりも高速に動作する．次に Matplotlib について説明する.Matplotlib は,Python およびその科学計算用ライブラリ NumPy のためのグラフ描画ライブラリである. オブジェクト指向の API を提供しており, 様々な種類のグラフを描画する能力を持つ. 描画できるのは主に 2 次元のプロットだが,3 次元プロットの機能も追加されてきている．自分は,gnuplot しか用いたことがなかったので最初にこのライブラリを使った時は便利すぎて驚いた. 卒論のグラフ画像などにも利用した．最後に Keras について説明する.Keras は,Python で書かれたオープンソースニューラルネットワークライブラリである.MXNet（英語版）,Deeplearning4j,TensorFlow,CNTK,Theano（英語版）の上部で動作することができる. ディープニューラルネットワークを用いた迅速な実験を可能にするよう設計され, 最小限, モジュール式, 拡張可能であることに重点が置かれている．Keras はどちらかというとライブラリというよりフレームワークの方が近い気がするが, 自分のような Python 初学者でもコードを記述しやすく理解しやすいライブラリである. 機械学習を組み込みたいときは上記の 3 つ Keras,Matplotlib,Numpy は必須だと思われる.

## 2 Makefile について

Makefile について, 講義で紹介した機能以外のものについて 2 つ以上調べて説明せよ.

- include
- SOURCES

まずは include から説明する. インクルードパスとして INCLUDE の値を用いる. 初期値は-I./include となっている．ソースファイル中の#include ファイル検索パスに加えるパスを-I オプションにて指定する.-I オプションとディレクトリ名の間に空白を書くことはできない．複数ディレクトリを指定したい場合は-I オプションを空白区切りで複数指定する．次に SOURCES について説明する. コンパイル対象となるソースファイルとして SOURCES の値を用いる。初期値は$(wildcard $(SRCDIR)/*.cpp)。 SRCDIR に存在する拡張子 cpp のファイル全てをコンパイル対象とすることを意味する。別の拡張子 (.c など) に変更したい場合は、makefile 内の cpp を全て変更する。

## 3 make 以外のビルドツールについて

1 つ以上調べて,make との違い・優れている点などを説明せよ.Autotools について調べたので説明する.Autotools とは, 主に Unix 系オペレーティングシステム (OS) においてソフトウェアパッケージ開発を行うための, ツール及び

フレームワークの一種である. このツールを使用することにより, 多種多様な UNIX 互換環境にパッケージを対応させることが容易になる. Autotools は主に autoconf/automake/libtools の 3 つから成り立っている. make と比べた利点としては

- 自動的に依存関係を生成できる.
- 複数の OS(プラットフォーム) をカバーしやすい.
- デフォルトで clean,install,dist などの標準的なターゲットが生成される.

などがあげられる.

## 4　サンプルプログラムの分割

サンプルプログラム (著者当てプログラム) について, 以下の機能を main から分離し, プログラムの分割をせよ.

- ファイル名と著者名などを記述した”database.csv”を読み込む関数
- 質問する関数
- 結果と作品名を表示する関数

それぞれ read_database.c,question.c,answer.c として分割し, それぞれ main.c で呼び出すようにした. 仕様としては,header.h を include して分割コンパイルできるようにした. 心残りがある点としては int i と int j を局所変数と定義したとき answer.c ではうまくいったが, なぜか read_database.c ではエラーがでてうまくいかなかった.for 文や while 文で用いる i や j を大域変数として定義するのはよくないとは思ったが色々試行錯誤したもののうまくいかなかったためそのままである.

## 5　makefile で一括コンパイル

第 4 節で分割したプログラムを makefile を用いて一括コンパイルできるようにせよ.

まずは普通に makefile を使用して一括コンパイルができるようにしたあとマクロ定義, サフィックスルールを用いて簡潔に書いた. 正直この量のプログラムではサフィックスルールを使用するメリットは感じられなかったが, もっと大規模なプログラムになると有用性がわかりそうだ感じた.

## 6　静的ライブラリ化

第 5 節にライブラリ化 (静的) を行い, リンクする記述を追加せよ.

サンプルファイルメイクファイルを参考に静的ライブラリ化を行った. 正直したことはマクロ定義の部分を変更しただけなのであまり理解度は高くないが, そういうものがあるという知識は得た.

## 7　動的ライブラリ化

第 5 節にライブラリ化 (動的) を行う記述を追加せよ.

サンプルメイクファイルを参考に動的ライブラリ化を行った. こちらもマクロ定義の部分を変更するだけだったが, 大域変数として定義した変数についてエラーメッセージがでてしまい, 試行錯誤したが解決しなかった.makefile 自体はかけていると思うが, もう少しライブラリ化については勉強する必要があると感じた.

## 8　より正確に著者を当てられるように改良せよ

こちらは時間が足りずにできなかった.