計算機言語 I: C の処理系をコンピュータに導入する

言語処理系の導入

プログラミングの勉強は, 実際にプログラムを作って動かす事がなければ, 実力が全くつきません.

自宅で使っている PC に何らかの C 言語の処理系を導入する方法を述べます.

1. 上級者向け

このページに書いてある内容は, PC に詳しい人向けです. 従って, 大雑把なことしか書いていません. この内容が理解できない人は, このページは無視して, 次ページ以降の事を実行してください.

上級者の方は, 次のいずれかができると思いますので, それを実行してください.

自分の PC に Linux や BSD 系の OS を導入して, エディタやコンパイラを整備する.

例えば, Raspberry Pi だと, 標準的な OS は Linux 系ですでに C の処理系が入ってます. 仮に compiler が入っていなくても, ネット経由で簡単に導入できます.

PC や Mac だと Virtual Box などの仮想マシンを導入するとか, Linux と本来の OS を起動時に切り替えるようにするとか, いろいろな方法があります. ネットで調べて, それを実行してください.

情報処理センターに login して実習する. すなわち, 自分の PC には処理系を導入しない.

情報処理センターのコンピュータ, cc.u-ryukyu.ac.jp は, 学外からも ssh で login 可能です. login の方法は, 計算機概論 I で述べましたので, 復習してください.

ssh の導入方法は OS によりますので, 各自調べてください. ssh の port forwarding による X Window システムの利用は, 通信回線の帯域幅からすると, 学内の Wifi 環境でないと, 現実的ではありません.

エディタは端末経由で, vim, emacs, nano 等を利用してください.

2. Mac の場合

この講義で用いる C compiler だけを導入するなら, Home brew `https://brew.sh/index_ja` を導入すれば, Mac 用の C compiler が自動的に Apple からダウンロードされます. エディタは, mi(`https://www.mimikaki.net`), CotEditor (`https://coteditor.com`) あたりが無料で評判の良いものです. 使い方は Linux と同様で, ターミナル (アプリケーション→ユーティリティ にある) が Linux の端末 (gnome-terminal) に対応します.

Home brew ではなく, macports でも同様だと思いますが, 使ったことが無いのでわかりません.

本格的に macOS や iOS の開発をするのなら, Apple から開発環境 Xcode をダウンロードします. Apple Store にアカウントを作って login し, ダウンロードします. 昔の OS を使っている場合も, その OS に対応するバージョンが Apple Store. にあります. (探すのが少し大変な感じ)

Xcode は iOS (iPhone, iPad) の開発環境も入ったすごく大きなシステムですので, それなりのネットワーク回線がないと, ダウンロードに時間がかかります. 大学のネットワークを利用すれば, 少しはマシかもしれません.

導入が終われば, 一度起動して, ライセンス条項に同意しておいてください. エディタは, Xcode に付属しています (最初のファイル作成で戸惑うのが難点). 使い方は, Xcode 内でコンパイル, 実行することもできますし, 上に書いたターミナル利用も可能です.

3. Windows の場合

まず, Windows update を実行して, 最新バージョンにします.

Windows は, Microsoft が無料の開発環境を公開するのが遅かったので, さまざまな処理系の選択肢があります. (今の MacOS X は, そもそも, ソフトウェア開発環境込みで作られていたので, 開発環境を無料で配布することが前提になっている部分がある.)

Microsoft Visual Studio

導入 (install) が最もやさしい C の処理系は, Microsoft が配布している, Visual Studio 2019 です. いくつかのエディションがありますが, Visual Studio Community は, 無料で導入できます. ただし, C の処理環境を入れるだけでも, Mac の Xcode と同じくらいのデータサイズになり, 全部を導入すると, とんでもない物体になります. なので, 必要とあれば, 大学のネットを利用してください. (Mac の項を参照)

「Visual Studio ダウンロード」で検索すれば, MicroSoft のダウンロードサイトに辿り着くと思います.

最初のダウンロードのソフトウェアの実行で, インストーラーのダウンロードが実行され, 次にインストーラーが実行されます. インストーラーを用いて導入するパッケージは,「C++ によるデスクトップ開発」のパッケージだけです.

最初の起動直後にサインインのウィンドウが現れますが, サインインは不要です.

次の選択画面では,「ローカルフォルダーを開く」を選んで, 左側にある, Visual Studio 2019 の所の「repos」を選んでください.

ソースファイルの作成時は,「ファイルメニュー → 新規作成 → ファイル」で Visual C++ を選んでおき, 保存の際に, C ソースを選びます.

- Visual Studio を起動した状態で,「ツール」メニューから「コマンドライン → 開発者用 Power Shell」が, Linux でいう Gnome 端末です.
- コンパイルコマンドは, `cl` (小文字のシーエル) です.
- できる実行可能ファイルのファイル名は, ソースファイルの`.c` が `.exe` に変わったものです. Linux と同様, `./hogehoge.exe` の形で実行します.
- Windows では, フォルダ階層は, ￥記号 (本来はバックスラッシュ \ ですが, 日本語の Windows だけ￥記号で表示される) です.
- ヘッダファイル (例えば, `#include` で利用される `stdio.h`) の場所とかが, Linux や Mac と大きく異なります. Windows のこれらについての解説は, 私はできません.

WSL(Windows Subsystem for Linux)

上の Visual Studio ですが, とんでもなく重い (処理が遅いことを PC 業界では重いという) 環境です.

昨年度の計算機概論 I で紹介した, Windows Subsystem for Linux を利用すると, グラフィカルではないものの, この講義と同じような環境は実現できます.

導入については, 昨年度の資料の WSL のところを読んでください.
`http://www.cc.u-ryukyu.ac.jp/~b977046/gairon/2020/02remote.pdf`

C コンパイラは,
`http://www.cc.u-ryukyu.ac.jp/~b977046/gairon/2020/wsl1.pdf`
に従って, gcc を導入します (`sudo apt-get install gcc`).

エディタも昨年紹介したサクラエディタを使います.
`https://sakura-editor.github.io`

WSL では, ホームフォルダは隠しファイルで, それを探すのは面倒です. Windows のホームフォルダが, WSL で見て, 次の絶対パスの場所にありますので, それを利用します (計算機概論の `cd` (change directory) を思い出す). Windows のユーザ名は, Windows でサインインしている時のユーザ名です.
`/mnt/c/Users/Windows のユーザ名`

C の処理系は, Visual Studio から比べると, かなり速く動きます.

WSL を用いて情報処理センターのようなグラフィカルな Linux 環境の構築は, 今の時点では, うまくできないようです. ある程度まではできるのですが, Web ブラウザがきちんと動かないとか, 終了後のキーボード設定が変になるとか, いろいろ問題が起きます.

その他

WSL 以外にも MinGW という C の処理系がありますが, これを利用するには, Windows Defender と戦う (というか戦いの相手から外してもらう) 必要があります.