PC 基礎講習テキスト
vol.1

谷口 陽音

2023-07-16

1　OS について

1.1　OS とは

OS[Operating System] は、一般に基本ソフトウェアのことを指し、これは PC が動作するために必要なものである。
OS にもさまざまな種類があり、さまざまなシステムに組み込まれる。
OS は主に C 言語で書かれている。また、OS によって実行ファイルの形式が異なる。
実行ファイルについては後述する。

1.2　OS の例

ここでは、代表的な OS を列挙する。

- PC
  - Windows
  - MacOS
  - ChromeOS
  - Linux[*1]
  - FreeBSD
- スマートフォン・タブレット
  - Android
  - iOS
  - iPadOS
  - Windows Phone
- サーバー
  - Linux
  - Windows Server

[*1] 詳しい種類 (ディストリビューション) については後述

表 1　主な OS と実行ファイル拡張子

OS	拡張子
Windows	.exe,.msi
MacOS	.app,.ipa
Android	.apk
iOS,iPadOS	.ipa
Linux	.appimage,.sh[*2]

1.3　実行ファイル

実行ファイルとは、その OS 上で実行可能なファイルのことを指す。
狭義には、アプリケーション・ソフトウェアの実体となるファイルを指す。
表 1 に代表的な OS とその実行ファイルの拡張子を示す。

1.4　Windows

Windows は、Microsoft 社が開発した OS で、クライアント向けの「Windows」シリーズとサーバー向けの「Windows Server」がある。
Windows Subsystem for Linux を用いることで Linux 環境を構築できる。
Windows のシェア率は非常に高く、70% を超えている。
なお、標準ブラウザーである Chromium 系の Microsoft Edge は今ひとつと言われている。

1.5　MacOS

MacOS は、Apple 社が開発した UNIX 系の OS で、Apple 純正コンピューター「Mac」シリーズ以外には搭載できない[*3]。
UNIX 系のため、Web デザインなどによく使われ、デザイン業界のスタンダードである。
iPhone や iPad といった他の Apple 製品と連携させることができる。
標準ブラウザーは Chromium 系の Safari であり、これも Apple 製 OS でないと動かない。

1.6　ChromeOS

ChromeOS は、Google が開発した Web ベースの OS である。
インターネット接続が前提とされた OS であり、RAM 容量および ROM 容量は少なくて良い。
純正 ChromeOS は Google Play Store で Android アプリのインストールが可能である。
なお、これのオープンソース版を Chromium OS という。

[*3] その MacOS を他のデバイスに移植しようとする Hackintosh という試みもあるが、これは Apple のライセンス違反となるため、推奨しない。

1.7 iOS/iPadOS

iOS および iPadOS は、Apple 社が開発したモバイル向け OS である。
これらの OS は、アプリストア「AppStore」経由でしかソフトウェアがインストールできない。
また、これらの OS 上で動作するアプリケーションについては、Mac でしか開発できない。

1.8 Android

Android は、Google が Linux をカスタマイズしてモバイルプラットフォームに最適化したモバイル向け OS である。
iOS/iPadOS とは違い、Google Play Store 以外のソースからのアプリもインストール可能である。
また、Android アプリは Mac だけでなく、Windows や Linux でも開発できる。
なお、Android アプリ開発用ソフト「Android Studio」は JetBrains の IntelliJ IDEA Community がベースになっており、開発言語もまた JetBrains 製の Kotlin が推奨されている。

1.9 Linux

1.9.1 Linux とは

Linux とは、伝説的エンジニア Linus Torvalds が Unix や Minix[*4]をもとに開発した OSS[*4]である。
今日の Web の普及は Linux があってこそのものだともいわれている。
また、先述したように、Google がソースコードを改変して Android を作ったように、OSS は改変や再配布が自由に可能である。
そもそも GUI を持っていないことも多いが、GUI を持っているものについては、デフォルトブラウザーは FireFox や Chromium といった OSS が採用されている。

1.9.2 Kernel と Distribution

狭義の Linux は「Kernel」と呼ばれる「核」にあたる部分のみを示す。Google が Android を作ったときにベースにしたのもこの Kernel である。
しかし、Kernel だけでは OS は動かず、他に Shell などのソフトウェアが必要で、Linus は GNU プロジェクトのソフトウェアを自力でインストールすることを前提としていた。
そこで、Kernel とその他動作に必要なソフト類をまとめて配布するパッケージが誕生した。これは、「Linux ディストリビューション (Distribution)」とよばれ、広義の Linux はこれを含むうえ、一般に Linux というと Linux ディストリビューションのことを指す。また、Linux ディストリビューションは、「ディストロ」と略されることも多い。いまでは、Linux ディストリビューションは星の数ほど存在し、日々新しいプロジェクトが立ち上がっては、プロジェクトが消滅していっている。

[*4] 当時よく使われていた教育向けのオープンソースな OS
[*4] Open Source Software: オープンソースソフトウェア (ソースコードを公開して開発すること)

表 2　ディストリビューションの分類

系統	特徴	パッケージマネージャー
Debian 系	最も多くのディストリビューションを擁する	apt
RedHat 系	元となったディストロは商用シェア No.1	rpm
Arch 系	安定性が高く、コミュニティが活発	pacman
独立系	上記の 3 つのいずれにも属さないディストロ	ディストロによる

1.9.3　Distribution の分類

現在ディストリビューションは星の数ほど存在し、分類をしなければ特徴を掴むのは極めて難しい。
よって、表 2 のような分類がある。なお、パッケージマネージャーについては後述する。

1.9.4　ユーザーインターフェース

ユーザーインターフェースは、略して UI ともよばれる、私達がコンピュータを操作するときにみる画面のことである。
Linux においては、GUI と CLI の 2 つに大別される。
GUI は、Graphical User Interface の略で、Windows や MacOS のように、キーボードとマウスなどを用いて操作できる形式である。
一方、CLI は Command Line Interface の略で、コマンド環境のみで操作する形式のことである。
CUI:Character User Interface,Character-based User Interface, または Command User Interface も CLI と同義である。

1.9.5　シェル

CLI 環境に置いては、コマンドの入力インターフェース兼コマンド入力補助ツールとして「シェル」とよばれるプログラムが使用される。
Windows では「コマンドプロンプト」や「Windows PowerShell」などに当たる。
おそらく一番有名なシェルは「Bash」である。

1.9.6　デスクトップ環境

GUI 環境をもつ Linux は、事実上デスクトップ環境 (略称:DE) を持つ。
表 3 に主なデスクトップ環境を挙げる。なお、谷口が今気に入っているのは KDE Plasma である。

1.9.7　Linux で利用するソフト

もちろん Linux でも十分実用に耐えうるようなソフトウェアが用意されている。
なお、この中には OSS であったり、OSS でなくても Windows や Mac でも利用可能なソフトが含まれる。このようなソフトウェアを「クロス・プラットフォーム」という。
あまり知名度が高くないソフトウェアも多いが、本プロジェクトでは基本的にこの中にあるようなソフトウェ

表 3　主なデスクトップ環境

DE 名	特徴	開発元
GNOME Shell	3D 効果を駆使したグラフィカルな DE	GNU
KDE Plasma	美しく、モダンで高機能性な DE	KDE
Xfce	軽さと機能性を両立したスマートな DE	
LXDE/LXQt	軽さを追求したとにかく軽い DE	
Budgie	すっきりとしたデザインでモダンな DE	

表 4　本プロジェクトで利用予定のソフトウェア

ソフト名	用途	類似ソフト
Inkscape	ベクター・グラフィックス	Adobe Illustrator
GIMP	画像編集	Adobe Photoshop
Krita	ラスター・グラフィックス	Clip Studio Paint
Microsoft Visual Studio Code	テキストエディタ	
GitHub Desktop	Git 管理	
Figma	UI 設計	

アを活用する。
今回活用する予定のクロス・プラットフォームのソフトを表 4 に掲載する。

1.9.8　コマンドライン・ツール

コマンド上から実行するツールのことをコマンドライン・ツールと言う。
私は Linux を常用している関係上、コマンドライン・ツールをよく利用する。
以下にその一例を挙げる

- Vim/NeoVim
- git
- GitHub CLI
- LilyPond
- Windows File Recovery
- LaTeX

1.10　まとめ

- OS はコンピュータの動作に必要不可欠
- 一口に OS といっても様々なものがある
- 実際の操作画面のことを UI といい、GUI と CUI という 2 種類に大別される
- 本プロジェクトでは基本的に OSS でクロスプラットフォームなアプリケーションを利用する

2　ソフトウェアについて

2.1　ソフトウェアとは

　ソフトウェアは、基本ソフトウェア (=OS) と応用ソフトウェア (=アプリケーション) に大別される。
基本ソフトウェアについては先述の通りであるため、ここでは応用ソフトウェアおよびソフトウェアの動作原理・プログラミングについて触れる。

2.2　ソフトウェアの構成と作成方法

　ソフトウェアは、プログラムとライブラリ・バッチファイルなどをまとめてパッケージ化して配布したものと捉えることができる。
プログラムは無論プログラミング言語を用いて作成される。大抵の場合次のような手順を踏む。

1. 計画
2. プログラミング
3. コンパイル
4. ビルド
5. 配布

以下では特にプログラミングに焦点を置く。

2.3　プログラミングとは

　プログラミングとは、コンピュータへの命令を書いたファイルを作成する一連の作業のことである。
ここで作成したファイルのことをプログラムと呼ぶ。
混同されがちなものにシェルスクリプトがあるが、シェルスクリプトはプログラムの 1 つと捉えることができる。

2.4　プログラミング言語

　プログラミングに用いる言語をプログラミング言語というが、これらは以下の 2 つに大別させる。

- コンパイラ言語
- スクリプト言語

それぞれに利点があるが、ここでは主にスクリプト言語を扱う

2.4.1　このプロジェクトで扱うプログラミング言語

　本プロジェクトは、主に Web を対象とするため、自ずと利用する言語は限られてくる。以下にそれぞれの特徴を示す

2.4.2 JavaScript

スクリプト言語で、主に Web 開発に使われる。
Web ブラウザ上で実行できるため、殆どの場合実行環境構築はしなくて良い。
なお、Node.js とよばれるパッケージを用いることで、ローカル環境でも実行でき、Electron というツールを使えば、デスクトップアプリも作成可能と、その活躍幅は Web だけに留まらない。

2.4.3 Python

スクリプト言語で、様々な用途に使われる。
Web 開発の中でもサーバーサイドの開発であるバックエンド開発によく用いられる。
AI などにも使われるため、身につけて損はないと思うが、インデントが意味を持つ珍しいタイプの言語である。

2.4.4 PHP

スクリプト言語で、Web 開発に使われる。
今回は JavaScript に統一する予定だが、もしかしたら使うかもしれない。
JavaScript とは違い、サーバーサイドで実行される。

2.4.5 HTML/CSS

正確にはプログラミング言語ではない。
HTML はマークアップ言語、CSS はスタイルシートである。
Web ページの枠組みや見た目を作るための言語である。

2.5 実際にアプリケーション作成によく使われている言語

OS によって大きく異なるため、OS ごとに記す

2.5.1 Windows

Windows のアプリ開発はわりと自由度が高いが、Microsoft 社が開発した C#を用いることが多い。
C#は C 言語系の Java および C++ の派生に位置付けられ、比較的シンプルに記述できる。
C#以外では、C++ や Visual Basic,Java,JavaScript などが用いられる。
なお、Windows 開発で用いる場合、C#,C++ などは Microsoft の「方言」があるため、Visual を付してよばれる。
また、C#,VB.NET,F#をまとめて.NET とよばれる。
IDE は Microsoft Visual Studio を使うことが多い。

2.5.2 MacOS,iOS,iPadOS

Apple 製品のアプリ開発は事実上 Swift または Objective-C 系統の 2 択である。(正確には Mac では JavaScript や Java も動く) また、Swift の標準 IDE は Mac 専用の「XCode」である。
Unity などで Windows でビルドはできるが、Mac でリビルドしなければ動かせない。

2.5.3 Android

Android は、少々前までは Java を使っていたが、Java の有償化に伴い、Kotlin が推奨されるようになった。
Kotlin はチェコの JetBrains 社が開発した言語である。
Android の標準 IDE である AndroidStudio は JetBrains 社の IntelliJ IDEA Community がベースになっている。
なお、Android Studio は Windows,Mac,Linux 対応である。

3 Markdown

本プロジェクトでは、標準ドキュメント形式として Markdown および LaTeX を採用する。
そのうち、みなさんには Markdown を活用できるようになっていただきたい。

3.1 Markdown とは

Markdown は、軽量マークアップ言語の 1 つで、GitHub や Qiita、Zenn などのエンジニア向けツールによく採用されている。
ただのプレーンテキストを少しの記号で見やすくすることができる。
これはメモとしても優秀で、構造化して記録できるため、非常に可読性が高いメモになる。

3.2 Markdown と HTML

Markdown は HTML と互換性があり、Markdown 内では HTML と Markdown の共存が可能である。
そもそも Markdown は HTML を簡略化したものであり、Markdown で実装されなかった HTML の機能を HTML のタグを用いて表現できる。

3.3 Mermaid

Markdown では、Mermaid 記法というものを用いて、図をかける。
一部のエディターは非対応であるが、多くのエディターで表示できる。

3.4 PlantUML

Markdown では、PlantUML を用いてシーケンス図などを作成できる。
しかし、Mermaid よりも対応しているものは控えめである。

3.5 記法

Markdown では、記号を用いて構造化する。

3.5.1 見出し

Markdown では、HTML 同様、第 1 レベル〜第 6 レベルの見出しを設定できる。

- # 第 1 レベル
- ## 第 2 レベル
- ### 第 3 レベル
- #### 第 4 レベル
- ##### 第 5 レベル
- ###### 第 6 レベル

3.5.2 箇条書き

Markdown では、よく使う記法である箇条書きが簡単にできる。

- - ハイフン
- * アスタリスク
- _ アンダーバー (アンダースコア)

のいずれかで箇条書きにできる

3.5.3 番号付きリスト

番号付きリストも作成できる

1. 1 つ目
2. 2 つ目
1. 3 つ目 番号は関係なく通し番号が振り直される

3.5.4 表

表は比較的簡単に作成可能| 列 1 | 列 2 | | — | :—: | | 内容 | 中央揃えの列 |

3.6 拡張子

Markdown ファイルの拡張子は [.md] または [.markdown] である。

4 コマンド操作と Git

ここでは、Windows PowerShell の使い方と Git について扱う

4.1 コマンドと引数

コマンド環境下に打ち込んで端末を制御する文字列をコマンドという。
また、コマンドのあとに文字列をつけることもあり、その文字のことを「引数」という。

読み方は「ひきすう」であり、これは数学の「因数」と区別するためにそうよばれる。

4.2 基本コマンド

PowerShell 上でのファイル操作コマンドについて説明する

4.2.1 ls

ls[list] コマンドは、今いるフォルダー [カレントディレクトリ] 内のファイルの一覧を表示するコマンドである。
引数にディレクトリパスを指定すれば、そのディレクトリの中身を表示する。

4.2.2 pwd

pwd[Print Working Directly] コマンドは、今いるディレクトリのパスを表示するコマンドである。
実行すると、現在のカレントディレクトリの絶対パスが出力される。

4.2.3 mkdir

mkdir[MaKe DIRectly] コマンドは、カレントディレクトリ以下にディレクトリを作成するコマンドである。
ディレクトリ名を引数に取る。

4.2.4 cd

cd[Change Directly] コマンドは、カレントディレクトリを移動するコマンドである。
移動先を引数に取り、パスは相対パスでも絶対パスでも可。

4.3 git のセットアップ

Git をインストールしたあとには少々設定が必要である。
そのときに使うのは Git コマンドだ。

4.3.1 git config

git config コマンドは、主に Git を使うための設定を行うコマンドである。
Git を使うために必要な設定コマンドをいかに示す

- git config –global user.email <email>
- git config –global user.name <name>

以上のコマンドは、Git を使用するにあたって、コミットする際に必要なユーザー名とメールアドレスを登録するコマンドである。

4.3.2 git clone

git clone コマンドは、リモートリポジトリをローカルにクローンするコマンドである。
Git で一番基本的なコマンドであり、始めるために必要不可欠なコマンドと言ってもいいだろう
構文を以下に挙げる。

- git clone <url>

ここで URL は、GitHub の場合、[https://github.com/<name>/<repo>] である。
たとえば、本プロジェクトのソースをクローンしたい場合、「git clone https://github.com/haru-0205/OMUCT_1-4」と入力する。

5 次回について

次回は、Markdown、今回のコマンドの続きとハードウェアについて行う。
特にコマンドでは、パッケージのインストールを行うコマンドを実際に実行してもらう。
それに伴い、(殆どの場合インストールされてると思うが、) 以下のソフトウェアを事前にインストールしておくこと。

- Windows Package Manager (WinGet)

Microsoft Store から簡単に導入できる。

なお、次回導入するソフトウェアは下記の通りである。
可能な限り導入は避けること。

- Obsidian
- GitHub CLI
- ImageMagic
- GitKraken