

PC基礎講習会

Vol.1

大阪公立大学工業高等専門学校 谷口 陽音

1. OSについて

1.1 OSとは

OS:コンピュータを使う上で必ず必要なソフトウェア

OSの役割

- コンピュータの入出力制御(=IO)
- メモリの管理
- ファイルの管理
- PCのパフォーマンスの向上のための制御
- API(プログラム用のインターフェース)の提供

■ OSにも様々な種類がある

■ OSごとに実行(可能)ファイルが異なる

1.2 OSの例

PC

- WIndows
- MacOS
- ChromeOS
- Linux

スマートフォン・タブレット

- Android
- iOS
- iPadOS

サーバー

- Linux
- Windows Server

1.3 実行ファイル

実行ファイル：そのOSで実行可能なファイル

シェルスクリプト ∈ 実行ファイル

ex:

OS	拡張子
Windows	.exe/.msi
MacOS	.app/.ipa
Android	.apk
iOS/iPadOS	.ipa
Linux	.appdata/.sh

1.4 Windows

項目	内容
開発元	Microsoft
種類	Windows,Windows Server
標準ブラウザ	Microsoft Edge
特徴	<ul style="list-style-type: none">■ Windows Subsystem for LinuxでLinux環境を構築可能■ シエア率が極めて高い(70%以上)■ MSEdgeはChromium系でChrome拡張が使える

1.5 MacOS

開発元

Apple

種類

MacOS

標準ブラウザ

Safari

特徴

- UNIX系でUNIXコマンドが実行可能
- Webなどに適している
- Appleデバイスとの連携が強い
- SafariはChromium系ではない

1.6 ChromeOS

開発元

Google

種類

ChromeOS/ChromeOS Flex/ChromiumOS

標準ブラウザ

Google Chrome

特徴

- Webベース
- リソース消費量が少ない
- (正式版のみ)Androidアプリが使える

1.7 iOS/iPadOS

開発元

Apple

種類

iOS/iPadOS

標準ブラウザ

Safari

特徴

- AppStore経由でしかアプリを入れられない
- アプリはMac以外では開発できない
- Apple製品との連携が強い

1.8 Android

開発元

Google

種類

Android/Android x86など

標準ブラウザ

Google Chrome/発売元独自ブラウザなど

特徴

- Linuxベース
- 野良アプリ(Google Play Store経由以外のアプリ)のインストールができる
- 開発環境はWin/Mac/Linuxに対応

1.9 Linux

1.9.1 Linuxとは

開発者 Linus Torvaldsおよびその他コミュニティ

種類 狹義にはKernel,広義にはDistribution(Distributionは極めて多種多様)

標準ブラウザ Firefox,Chromiumなど様々

- 「伝説的エンジニア」Linus Torvaldsが大学生のときに開発
- 昨今のWeb社会の構築に一役買っている
- OSS

1.9.2 KernelとDistribution

Kernel

OSの核

Distribution

様々な周辺ソフトを添付して配布されたもの

Kernel

- OSの核にあたる部分
- 狹義のLinux
- Shellなどは自力インストールをすることが前提

Distribution

- LinuxはKernelのみでは動かない
- コミュニティによって開発
- 略して「ディストロ」

1.9.3 Distribution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ロを擁する	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つのいずれにも該当しない	ディストロによる

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(=UI):コンピュータを扱うときに見る画面

- GUIとCLI
- GUI(Graphical User Interface)
 - キーボード&マウス
- CLI(Command Line Interface)=CUI(Character User Interface/Character-based User Interface/Command User Interface)
 - キーボードのみ
 - Shellを使う

1.9.5 シェル

シェル:コマンドの入力インターフェース兼コマンドの入力補助プログラム

- WindowsのコマンドプロンプトやPowerShellにあたる
- Linuxでは通常Bash

1.9.6 デスクトップ環境

デスクトップ環境(Desktop Environment = DE):GUIの環境

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

1.9.7 Linuxで利用するソフト

Linuxでも十分実用的なソフトがある
=>クロス・プラットフォーム

ソフトウェア名	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	

1.9.8 コマンドライン・ツール

コマンドライン・ツール:コマンド上から実行するツール

- 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不可欠
- 一口にOSといっても様々な種類がある
- UIはGUIとCUIの2つに大別される
- 本プロジェクトでは基本的にOSSかつプラットフォームを利用する

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(中学技術)
- 基本ソフトウェア(OS)
- 応用ソフトウェア(アプリケーション)

ここでは、応用ソフトウェアにフォーカスする

2.2 ソフトウェアの構成と作成方法

- ソフトウェア=プログラム+ライブラリ+バイナリファイル+...
- 次の手順で作成する
- 計画
- プログラミング
- コンパイル
- ビルド
- 配布

以下ではプログラミングにフォーカスする

2.3 プログラミングとは

- プログラミング=コンピュータへの命令を書いたファイルの作成
- シェルスクリプト ∈ プログラム

2.4 プログラミング言語

プログラミング言語

- コンパイラ言語
- スクリプト言語

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Web系だから自ずと利用言語は限られる

2.4.2 JavaScript

- スクリプト言語
- 主にWeb
- Node.JS,ElectronでPC向けアプリも作れる

```
//sample.js
let age = 15;
Console.log("Hello,World!")
if age <= 18{
    Alert("成人です")
}else{
    Alert("未成年です")
};
document.getElementById("AGE").innerText = age;
```

2.4.3 Python

- スクリプト言語
- 様々なことに使われる
 - サーバーサイドプログラム
 - AI
- インテントが意味を持つ

```
#sample.py
age = 15;
print("Hello,World")
if age <= 18:
    print("成人です")
else:
    print("未成年です")
print(age)
```

2.4.4 PHP

- スクリプト言語
- Web開発
- 今回はJSに統一予定
- サーバーサイド

```
//sample.php
$age = 15;
echo 'Hello, World!';
if ($age >= 18){
    echo "成人です";
} else{
    echo "未成年です";
}
```

2.4.5 HTML/CSS

- Webページの見た目と要素を設定
- 正確にはプログラミング言語ではない
- HTML(Hyper Text Markup Language):マークアップ言語
- CSS(Cascading Style Sheets):スタイルシート

```
<!--sample.html-->
<!DOCTYPE html>
<html>
  <head>
    <meta charset=utf-8>
    <title>sample page</title>
    <link rel="stylesheet" href="sample.css">
  </head>
  <body>
    <p class=midashi>Hello,World!</p>
    <p class=age>年齢:<span id="AGE"></span></p>
    <script src="sample.js"></script>
  </body>
</html>
```

```
/*sample.  
css*/  
.midashi{  
    font-size: 20pt;  
    color: navy;  
    border-bottom: 2px solid;  
}  
.age{  
    font-size: 18pt;  
    color: red;  
    border-bottom: 2px dashed;  
}  
#AGE{  
    font-size:22pt  
}
```

2.5 実際にアプリケーション作成によくつかわれ q 言語

OSによって異なるためOSごとに記述する

2.5.1 Windows

- 自由度はわりと高い
- 主にMicrosoft製のC#が使われる
- C#はC言語系でC++およびJavaの派生
- C#以外だとVisual Basic, Java, JavaScript, C++など
- C#, C++, VBなどは方言がある

```
//sample.cs
using system;
public static void main{
    int age = 15;
    console.WriteLine("Hello,World");
    if(age <= 18){
        console.WriteLine("成人です");
    } else{
        console.WriteLine("未成年です");
    };
    console.WriteLine(age);
}
```

2.5.2 MacOS,iOS,iPadOS

- 原則SwiftまたはObjective-c系
- IDEはXCode
- Macでしか開発できない(Macは例外的にJavaも動かせる)
- UnityなどでWindowsでビルドできるがMacでリビルドしなければ動かない

```
//sample.swift
var age: Int = 15;
print("Hello,World!");
if age >= 18
    print("成人です");
else
    print("未成年です");
print(age);
```

2.5.3 Android

- Kotlinが推奨されている
- 少し前まではJava
- IDEのAndroid StudioはJetBrains IntelliJ IDEA Communityベース
- Android Studioはクロスプラットフォーム

```
//sample.kt
fun main (args:string){
    int age = 15;
    println("Hello,World!")
    if (age >= 18){
        println("成人です")
    }else{
        println("未成年です")
    }
}
```

3. Markdown

はじめに

本プロジェクトの標準ドキュメント形式

- Markdown
- LaTeX

よって、皆さんにはMarkdownを扱えるようになってもらう

3.1 Markdownとは

- 軽量マークアップ言語の一つ
- Markdownの活用例
- エンジニア向けツール
 - Qiita
 - Zenn
 - GitHub
- プレーンテキストを少しの記号で見やすくする
- メモとして使用する

3.2 MarkdownとHTML

- MarkdownとHTML間には互換性がある
- そもそもMarkdownはHTMLを簡略化したもの
- Markdownで実装されなかった部分はHTMLタグで記述

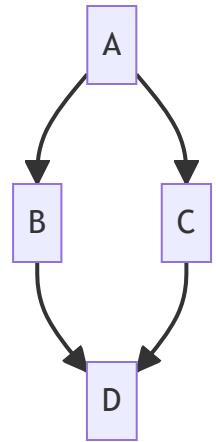
Markdown<=>HTMLの互換性は高い

3.3 Mermaid

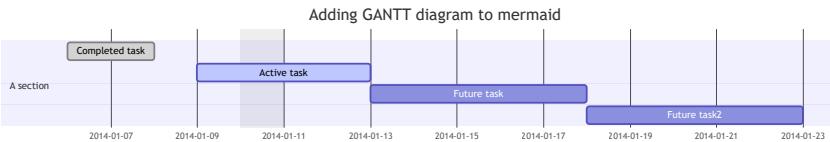
一部のMarkdownエンジンではMermaidを使える
Mermaidは以下のような図をかけるエンジン

- フローチャート
- シーケンス図
- ガントチャート
- クラス図
- Gitグラフ
- ER図
- ジャーニーマップ
- クアドラントチャート

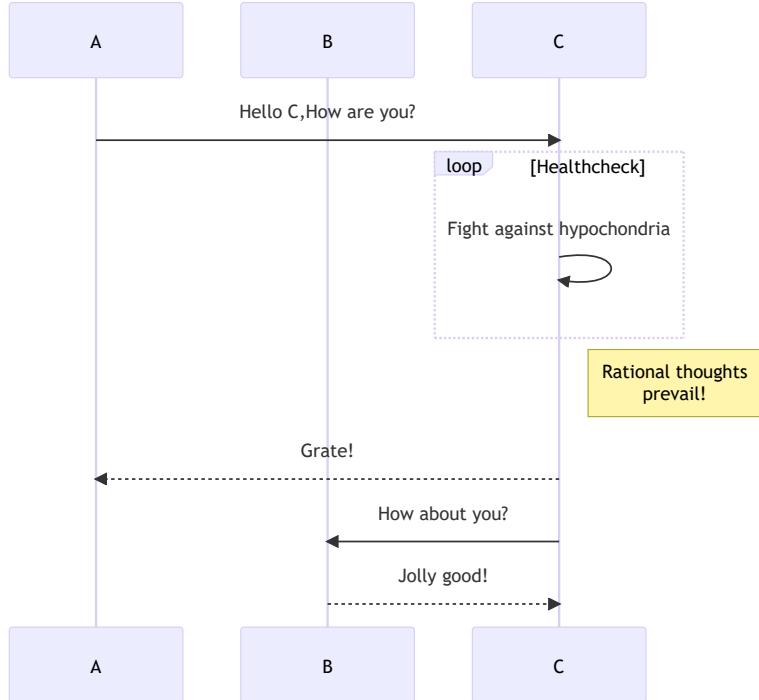
フローチャート



ガントチャート



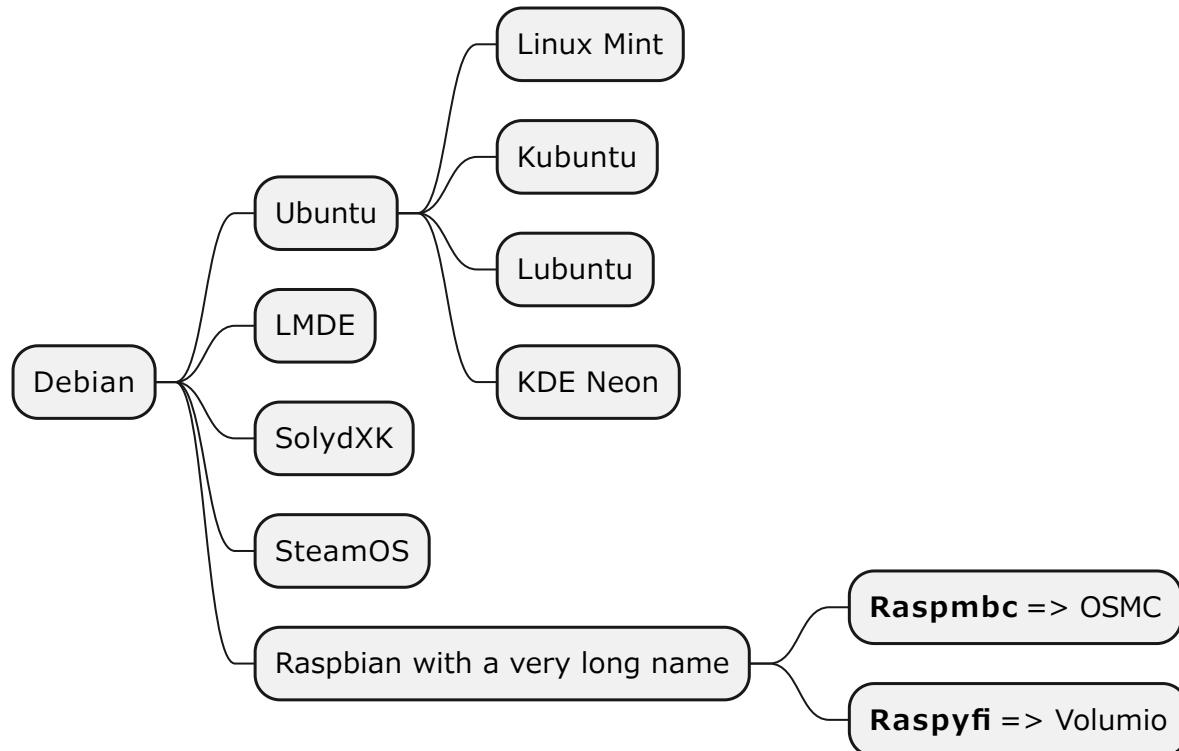
クラス図



3.4 PlantUML

- Mermaidの様なグラフ作成ツール
- シーケンス図
- ユースケース図
- クラス図
- オブジェクト図
- アクティビティ図
- コンポーネント図
- 配置図
- 状態遷移図
- タイミング図
- ガントチャート
- マインドマップ
- ER図
- 数式

マインドマップ



3.5 記法

Markdown=>記号を用いて構造化

3.5.1 見出し

- 第1レベル(大)~第6レベル(小)
- HTMLとの互換性あり

```
# 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
```

- 出力

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

(本文)

3.5.2 箇条書き

非常に簡単

- 箇条書き
- * これでも可
- + これもOK
 - ネストも可能

- 箇条書き
- これでも可
- これもOK
 - ネストも可能

3.5.3 番号付きリスト

番号付きリストも可能

- 1. 1つ目
- 2. 2つ目
- 1. 番号は何でもいい
 - 1. ネストも可能
 - 3. 番号がとんでもOK

- 1. 1つ目
- 2. 2つ目
- 3. 番号は何でもいい
 - 1. ネストも可能
 - 2. 番号がとんでもOK

3.5.4 表

表も割りと簡単に作れる

| 列1 | 列2 | 列3 | 列4 |

| --- | :--- | :---: | ---: |
| デフォルト | 左詰め | 中揃え | 右詰め |
| 1 | 2 | 3 | 4 |

列1	列2	列3	列4
デフォルト	左詰め	中揃え	右詰め
1	2	3	4

3.6 拡張子

- Markdownの拡張子は以下の2種類
- .md
- .markdown

4. コマンド操作とGit

4.1 コマンドと引数

- コマンド:特定の文字列を利用してPCを制御する
- コマンドのあとにつける対象などの文字列を**引数**という
- 引数の読み方はひきすう(数学の因数と区別するため)
- コマンドのあとにつける文字列には以下の種類がある
- 引数
- サブコマンド
- コマンドオプション

4.2 基本コマンド

PowerShell上、およびMac,Linuxで使用できるコマンド

4.2.1 ls

- ls:カレントディレクトリ以下のファイル・ディレクトリを一覧表示
- LiSt
- 対象ディレクトリを引数にとることができる

4.2.2 pwd

- pwd:カレントディレクトリのパスを表示
- Print Working Directory

4.2.3 mkdir

- mkdir:カレントディレクトリ直下にディレクトリ作成
- MaKe DIRectory
- 作成先ディレクトリを引数にとることができる

4.2.4 cd

- cd:カレントディレクトリの移動
- Change Directory
- 移動先ディレクトリを引数にとる

4.3 gitのセットアップ

4.3.1 git config

Gitをインストールしたあとには設定が必要
以下にそのときのGitコマンドを示す

```
git config --global user.email <mail>
git config --global user.name <name>
```

以上のコマンドの役割 コミットする際に必要な以下の情報の設定

- ユーザー名
- メールアドレス

4.3.2 git clone

- リモートリポジトリをローカルリポジトリにクローンする
- Gitで一番基本的
- Gitを始めるために必要

```
git clone <URL>
git clone https://github.com/haru-0205/markdown
```

5. 次回について

次回は以下のことを行う

- 今回のコマンドの続き(実際にいくつかコマンドを実行)
- Markdownの続き
- ハードウェアについて

それにともない、Wingetが必要だからインストールしておくこと。

次回導入するソフトウェア(まだインストールしなくてよい)

- Obsidian
- GitHub CLI
- ImageMagic
- GitKraken

また、次回の講習ではGitHubアカウントが必要だから準備しておくこと