

The primary objective of this document is to facilitate a conceptual comprehension of “What constitutes system development” for individuals lacking IT skills and industry experience. This endeavor is not aimed at presenting efficient methodologies or challenging widely accepted norms; rather, it seeks to methodically consolidate information for those desiring a fundamental grasp of essential points.

Contents

I Console	3
I -1 コンソール	3
I -2 ターミナルとシェル	3
I -3 コマンド	3
I -4 パイプとリダイレクト	4
I -5 正規表現	4
II Coding	4
II -1 コーディング	4
II -2 Hello, World!	4
II -3 変数	4
II -4 型	4
II -5 関数	4
II -6 コメント	4
II -7 制御フロー	4
II -8 所有権	4
III Algorithm	4
III -1 バブルソート	4
III -2 選択ソート	4
III -3 挿入ソート	4
III -4 シェルソート	4
III -5 バケットソート	4
III -6 分布数え上げソート	4
III -7 基数ソート	4
III -8 マージソート	4
III -9 ヒープソート	4
III -10 クイックソート	5
III -11 二分探索	5
III -12 フィボナッチ数列	5
III -13 線形探索	5
III -14 三文探索	5
III -15 N-Queen 問題	5
III -16 数独	5
III -17 最大和問題	5
III -18 ナップサック問題	5
III -19 部分和问题	5
III -20 最小個数部分和问题	5
III -21 最長共通部分列問題 (Longest Common Subsequence)	5
III -22 レーベンシュタイン距離	5
IV Quality Control	5
IV -1 単体テスト	5
IV -2 結合テスト	5
IV -3 総合テスト	5
IV -4 テスト 駆動開発	5
V Git	5
V -1 分散型バージョン管理システム	5
V -2 Git の仕組み	5
V -3 ブランチの運用	5
VI Entity Component System	5
VII Web	5
VII -1 沿革	5
VII -2 JavaScript	5
VII -3 JavaScript エンジン	5

VII -4 JavaScript ランタイム	6
VII -5 JavaScript を動かしてみる	6
VII -6 DOM	6
VIII Database	6
VIII -1 データベースとは	6
VIII -2 SQL	6
IX Technical Writing	6
X Typst	6
Bibliography	6
APPENDIX A: Foo	6
APPENDIX B: Bar	6

I Console

I -1 コンソール

コンソールとは、OS を搭載したコンピュータに接続されたディスプレイおよびキーボードである。コンピュータとの対話を可能にするための物理ハードウェア、入出力システムともいえる。

「文字だけの黒い画面」というキーワードで一般に想起される画面は、仮想コンソールである。これは入出力を仮想的な空間に提供するものであり、この環境は CUI (Character-based User Interface) ともいう。CUI は、我々が日々利用する GUI (Graphical User Interface) と対をなす存在であり、文字に基づいたユーザーインターフェースを提供する。

インターフェース

“Interface”という語はハードウェアでもソフトウェアでも登場する。主に情報産業では、異なる二つのモノ(人間や機器)を接触させるための境界面として用いられる語である。

I -2 ターミナルとシェル

仮想コンソール (CUI) を提供するアプリケーションは一般に**ターミナル**と呼ばれる。Windows におけるコマンドプロンプトや、MacOS における iTerm2 や Warp などがターミナルにあたる。これらは GUI 上で CUI を操作するための窓口になる。

本題となるコンピュータへの命令は**コマンド**という。コマンドを解釈し、実行する仕組みをシェルという。シェルの役割はコンピュータとの対話を実際に担うアプリケーションであり、Bash や zsh, fish など種類が存在する。

シェル

エンジニアでない一般ユーザが OS と対話するためのシェルに、Windows のエクスプローラーなどがある。CUI のシェルと GUI のシェル、操作しやすいユーザ層は当然異なる。

要するに、エンジニアがコンピュータへ命令するためには基本的に「ターミナルを起動し(シェルを介して)コマンドを入力」する。また、コマンドを入力する行のことをコマンドラインという。単語が重要なのではなく、この CUI 構成を理解することは今後役に立つ。

I -3 コマンド

ここでは、基本的なコマンドを紹介する。細かいオプションまでは解説をしない。なお、ここで使用するコマンドは Windows コマンドプロンプトでは使用できない。

I -3 -1 ファイルの一覧化 : ls

まずはフォルダの内容を確認したい。そのようなときに打つコマンドである。何回打っても良い。

```
1 ls
```

Command

I -3 -2 ディレクトリの移動 : cd

今いるフォルダから移動したいときに打つ。

```
1 cd {path} // {移動先のパス}へ移動 Command
2 cd .. // 1つ上のディレクトリへ移動
3 cd ~ // ホームへ移動
```

コマンドライン

本書では、コマンドラインへの入力表記において {} を代入記号、// をコメントとする。これらは実際に入力せず、可読性を補助する。

I -3 -3 現在地 : pwd

プロンプトに書いてあるディレクトリ名をわざわざ絶対パスで表示する。あまり使わないが、重大な作業をしているときに使った記憶がある。

```
1 pwd
```

Command

I -3 -4 ディレクトリの作成 : mkdir

思ったよりお世話になる。メイクディレクトリと読んでしまうが正解は知らない。知っておいて損はない。

```
1 mkdir {ディレクトリの名前}
```

Command

I -3 -5 ファイルの閲覧 : cat

本来は複数のファイルを結合(concat)して表示するコマンドだが、単一のファイルをのぞき見することで使うことが多い。

```
1 cat {ファイル名}
```

Command

I -3 -6 ページャ : less

本来のファイル閲覧用コマンドだが、大体は cat コマンドで足りてしまう。less を使用した後は、q キーを押下すると終了する。

```
1 less {ファイル名}
```

Command

I -3 -7 容量の確認 : du

「このフォルダは何 MB あるのか」といった疑問を解決する。ちなみに、`-hs` オプションを付与すると内訳を非表示にできるため、ファイル数が異常に多いディレクトリで有効である。

```
1 du {フォルダ名}
```

Command

I -3 -8 検索 : grep

ファイルを検索することは多々ある。よく使うオプションつきで紹介する。

```
1 grep -rn {path} -e ${text}
```

Command

- `r` : 再帰的にサブディレクトリを含める
- `n` : マッチした行番号を表示
- `e` : 検索する文字列を指定

I -3 -9 改名もしくは移動 : mv

便利だが、個人的にリスクがあると思っているコマンドの1つである。2つめの引数が存在するパスならそこへ1つ目の引数に指定されたファイルを移動させる。存在しないパスなら、1つ目の引数に指定されたファイルを2つ目の文字列に改名する。

```
1 mv {path} {path} // 改名
```

Command

```
2 mv {path} {rename} // 新しい名前に変更
```

移動先を誤字脱字すると、その文字列が新しい名前となるので注意が必要である。ただし、`-iv` オプションを付与すると警告してくれるので必須である。

I -3 -10 削除 : rm

`mv` より危険なコマンドである。削除をコマンドで実施する場合、取り返しがつかない(=ゴミ箱に移動しない)ため、`-iv` オプションを付与して警告を有りにする習慣をつけよう。

```
1 rm -iv {対象ファイル}
```

Command

I -4 パイプとリダイレクト

I -5 正規表現

I -5 -1 ワイルドカード

II Coding

II -1 コーディング

プログラムを開発する作業全般をプログラミングといい、特にソースコードを書く工程をコーディングという。コーディングする上でシンタックスとデザインパターンは重要なコアである。

シンタックスとは、プログラミング言語の仕様として定められた構文規則を指す。デザインパターンについてはWiki¹を引用する。

ソフトウェア開発におけるデザインパターンまたは設計パターン(英: design pattern)とは、過去のソフトウェア設計者が発見し編み出した設計ノウハウを蓄積し、名前をつけ、再利用しやすいように特定の規約に従ってカタログ化したものである。

・ Wikipedia

II -2 Hello, World!

ハローワールドプログラムを観察する。

```
1 fn main() {
2     println!("Hello, world!");
3 }
```

Rust

II -3 変数

II -4 型

II -5 関数

II -6 コメント

II -7 制御フロー

II -7 -1 if

II -7 -2 loop

II -7 -3 while

II -7 -4 for

II -8 所有権

III Algorithm

III -1 バブルソート

III -2 選択ソート

III -3 挿入ソート

III -4 シェルソート

III -5 バケットソート

III -6 分布数え上げソート

III -7 基数ソート

III -8 マージソート

III -9 ヒープソート

¹<https://w.wiki/rvm>

III -10 クイックソート

III -11 二分探索

III -12 フィボナッチ数列

III -13 線形探索

III -14 三文探索

III -15 N-Queen 問題

III -16 数独

III -17 最大和問題

III -18 ナップサック問題

III -19 部分和問題

III -20 最小個数部分和問題

III -21 最長共通部分列問題 (Longest Common Subsequence)

III -22 レーベンシュタイン距離

IV Quality Control

IV -1 単体テスト

IV -2 結合テスト

IV -3 総合テスト

IV -4 テスト駆動開発

V Git

V -1 分散型バージョン管理システム

V -1 -1 Git と GitHub

V -2 Git の仕組み

V -2 -1 ワーキングディレクトリ

V -2 -2 ステージング

V -2 -3 リポジトリ

V -2 -4 コミット

V -2 -5 プッシュ

V -2 -6 プル

V -2 -7 ブランチ

V -3 ブランチの運用

VI Entity Component System

VII Web

VII -1 沿革

本格的にワールドワイドウェブ (www) が普及したのは 1995 年頃だった。 当時は静的な HTML (HyperText Markup Language) ページが主流だった。 HTML は情報を構造化し、文書の意味や見出し、段落、リストなどを表現していた。 ページのデザインやスタイルは限定的だったが、これが Web の基盤となり、今日の進化したウェブページの基礎となった。

ユーザはインターネットブラウザを介して Web サーバと通信し、HTML を取得している。 さらに、CSS や JavaScript などがページデザインを制御することにより、インタラクティブな体験を得られている。

VII -2 JavaScript

JavaScript は 1995 年に Brendan Eich 氏によって開発された。 クライアントサイド²で動作するスクリプト言語として誕生し、HTML に動きや対話性をもたらした。

HTML で定義された要素に対し、イベントや条件に応じて操作や変更を行うことも機能の一つである。 例えば、ボタンをクリックしたときにアラートを表示したり、入力フォームの値を検証したり、画像を切り替えたり、アニメーションやゲームを作ることにも可能である。 JavaScript ライブラリ³を利用して高度な Web アプリケーションも開発されている。

著名な SNS もすべて、ブラウザ上では JavaScript が動いている。

VII -3 JavaScript エンジン

JavaScript コードを解析するコアプログラムを JavaScript エンジンという。 Web ブラウザはエンジンを搭載されているため、JavaScript を実行できる。

Browser	Engine
Google Chrome	V8
Microsoft Edge	V8
FireFox	SpiderMonkey
Safari	JavaScriptCore

JavaScript エンジン独自に再開発するプロジェクト⁴も存在する。

²サーバではなくユーザー側のモバイルや PC 上を指す。

³流行り廃りは <https://stateofjs.com> から確認できる。

⁴<https://github.com/boa-dev/boa>

VII -4 JavaScript ランタイム

JavaScript エンジンを含む、JavaScript 実行環境をランタイムという。

- Node.js(V8 を採用)
- Bun.sh(JavaScriptCore を採用)
- Deno(V8 を採用)

VII -5 JavaScript を動かしてみる

任意のディレクトリで新規の HTML ファイルを作成する。

```
1 touchn index.html
```

```
sh
```

以下はいわゆるトースト通知を実装したスクリプトである。このままコピー & ペーストし、index.html を Web ブラウザで開くと JavaScript の挙動を確認できる。JavaScript が埋め込まれた HTML ファイルは、特別なコマンドや操作は必要とせず、Web ブラウザに自動的に実行される。

```
1 <!DOCTYPE html>
2 <html lang="jp">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport"
6       content="width=device-width, initial-
7       scale=1.0" />
8     <title>Exaple</title>
9     <style>
10      #toast {
11        display: flex;
12        visibility: hidden;
13        height: 60px;
14        width: 300px;
15        border: 1px solid rgba(0, 0,
16          0, 0.1);
17        position: absolute;
18        top: 50%;
19        left: 50%;
20        transform: translate(-50%,
21          -50%);
22      }
23      #toast > p {
24        margin: auto 8px;
25        font-size: 14px;
26        color: #155724;
27        letter-spacing: 0;
28        line-height: 18px;
29      }
30    </style>
31  </head>
32  <body>
33    <div id="toast">
34      <p>Hello, JavaScript!</p>
```

```
31 </div>
32   <button class="button">Enter</
33   button>
34   <script>
35     document.addEventListener("DOMConte
36       function () {
37         const toast =
38           document.querySelector("#toast");
39         const button =
40           document.querySelector(".button");
41         button.addEventListener("click",
42           () => {
43             toast.style.visibility =
44               "visible";
45             setTimeout(function () {
46               toast.style.visibility
47                 = "hidden";
48             }, 3000);
49           });
50       });
51   </script>
52 </body>
53 </html>
```

<script>タグで囲まれている箇所が JavaScript にあたる。本書では、JavaScript のシンタックスについては触れないが、Coding セクションが理解できていれば問題ない。既に Rust がプログラミング言語の中で母国語のような存在に昇格できていれば、おそらく JavaScript もある一定レベルまで読むことができるだろう。

また、上記の HTML には最低限の CSS によってスタイリングされている。削除した場合にどのようなふるまいになるか、確認してみるといい。

VII -6 DOM

VIII Database

VIII -1 データベースとは

VIII -2 SQL

IX Technical Writing

X Typst

Bibliography

APPENDIX A: Foo

APPENDIX B: Bar