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

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
2 cd ..     //  1つ上のディレクトリへ移動
3 cd ~      //  ホームへ移動
```
Command

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． コーディングにおいて，シンタックスとデザインパターンは中核を成す．

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 Wiki[1] 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

これより，シンタックスに関連する事項を説明する． 通常，伝統的なアプローチでは「ハローワールド関数」から始めることが一般的だが，本書では変数に関する説明から始める．

II -2 変数

以下のコードはどのような処理か考えてみよう．

```
1 let x = 4;                                         Rust
```

x は変数，4 は整数リテラル[2] である． それと同時に，x は 4 を束縛 (bind) する所有者である． let は変数宣言のキーワードとなっており，以下の構文を想定している．

```
1 let PATTERN = EXPRESSION;                          Rust
```

整数リテラル 4 単体であっても，右辺は式である． 式が返却する値は 4 であり，左辺のパターン x に当てはめて束縛される．

パターンであるという認識はとても重要で，以下はエラーにならない．

```
1 let (a, b) = (4, 5);                               Rust
```

ただし，パターンには 2 つの形式が存在する． 上記の _Irrefutable(論駁不可能)なパターンと，*let 式で扱うことができない _Refutable*(論駁可能)なパターンである． この _Refutability(*論駁可能性*)の概念については，*エラーメッセージで見かけた際に対応できるように，慣れておく必要がある*． Refutable_ なパターンはまた別の章で触れる．

変数宣言は，値 4 に別のラベル x を張り替えるような作業ではなく，所有者 x に 4 が束縛されるという認識をもってほしい．

[1] https://w.wiki/rvm

[2] リテラルとはソースコードに生で書かれたデータであり，値そのものである．例えば，4(リテラル)に 5(値) を束縛することはできない．

II -3 型

Rust 標準のプリミティブ型を表にした． 型の名前に含まれる数字は，ポインタのサイズ (ビット)を指す．

基本データ型	型
符号付き整数	isize, i8, i16, i32, i64
符号無し整数	usize, u8, u16, u32, u64
浮動小数点数	f32, f64
文　字 (Unicode のスカラー値)	char

特に数値を表す型の許容値は以下となる．

型	MIN	MAX
i8	-128	127
i16	-32768	32767
i32	-2^{32}	$2^{32}-1$
i64	-2^{64}	$-2^{64}-1$
u8	0	255
u16	0	65535
u32	0	2^{32}
u64	0	2^{64}
f32	凡そ -21 億	凡そ +21 億
f64	凡そ -9 京	凡そ +9 京

isize および usize は実行環境 PC の CPU のビット数によって適切なサイズに変化する．

II -3 -1 型推論

初期化の際に評価値の型をチェックするだけでなく，その後にどのような使われ方をしているかを見て推論する．

II -4 関数

ハローワールドプログラムを観察する．

```rust
fn main() {
    println!("Hello, world!");
}
```

II -5 コメント

II -6 制御フロー

II -6 -1 if

II -6 -2 loop

II -6 -3 while

II -6 -4 for

II -7 所有権

II -8 非同期処理

II -9 構造体

II -10 Trait

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

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

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[3]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も可能である． JavaScript ライブラリ[4]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 エンジンを独自に再開発するプロジェクト[5]も存在する．

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．

```sh
toucn index.html
```

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．

```HTML
<!DOCTYPE html>
<html lang="jp">
  <head>
    <meta charset="UTF-8" />
```

[3]サーバではなくユーザー側のモバイルや PC 上を指す．

[4]流行り廃りは https://stateofjs.com から確認できる．

[5]https://github.com/boa-dev/boa

```
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Exaple</title>
7     <style>
8       #toast {
9         display: flex;
10        visibility: hidden;
11        height: 60px;
12        width: 300px;
13        border: 1px solid rgba(0, 0, 0, 0.1);
14        position: absolute;
15        top: 50%;
16        left: 50%;
17        transform: translate(-50%, -50%);
18      }
19      #toast > p {
20        margin: auto 8px;
21        font-size: 14px;
22        color: #155724;
23        letter-spacing: 0;
24        line-height: 18px;
25      }
26    </style>
27  </head>
28  <body>
29    <div id="toast">
30      <p>Hello, JavaScript!</p>
31    </div>
32    <button class="button">Enter</button>
33
34    <script>
35    document.addEventListener("DOMConte function () {
36      const toast = document.querySelector("#toast");
37      const button = document.querySelector(".button");
38
39      button.addEventListener("click", () => {
40        toast.style.visibility = "visible";
41        setTimeout(function () {
42          toast.style.visibility = "hidden";
43        }, 3000);
44      });
45    });
46    </script>
47  </body>
48 </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

VII -7 Web アセンブリ

VIII Database

VIII -1 データベースとは

VIII -2 SQL

VIII -3 NoSQL

IX Network Infrastructure

IX -1 インターネットとネットワークの定義

ネットワークとはコンピュータとコンピュータが繋がることで情報を通信できる仕組みであり，構造を指す．外部ネットワーク同士が繋がりあい，結果的に形成された地球上を駆け巡る巨大なネットワークの総称がインターネットである．

IX -2 インターネットの仕組み

総務省を引用する．[1]

> ネットワーク上で，情報やサービスを他のコンピュータに提供するコンピュータをサーバ，サーバから提供された情報やサービスを利用するコンピュータをクライアントと呼びます．私たちが普段使うパソコンや携帯電話，スマートフォンなどは，クライアントにあたります．
>
> · 総務省

IX -2 -1 IP アドレス

IPアドレスは，コンピューターやスマートフォンなどがインターネット上で通信するときに使われる，そのデバイスを特定するための「住所」のようなものである．家の住所があるように，コンピューターもそれぞれが異なるIPアドレスを持っている．これにより，データがどのデバイスに届くかが決まる．

IX -2 -2 DNS

DNS は，Domain Name System（ドメインネームシステム）の略であり，人間が覚えやすいウェブサイトの名前（例: www.google.com ）を，コンピューターが理解しやすい IP アドレスに変換するシステムである． 言い換えると，DNS はインターネット上での住所録のようなものです． あるウェブサイトにアクセスするとき，DNS がそのウェブサイトの名前を IP アドレスに変換してくれます．

IX -2 -3 サブネットマスク

サブネットマスクは，IP アドレスをグループ分けする． これにより，同じネットワーク内のデバイス同士は通信しやすくなる． IP アドレスは通常， ネットワーク部とホスト部に分かれており，サブネットマスクは，どの部分がネットワークで， どの部分がホストであるかを示す． ネットワークを効果的に管理し， 通信を効率的に行う目的がある．

IX -2 -4 ゲートウェイ

ゲートウェイは， コンピューターネットワークにおいて異なるプロトコルやネットワーク間で通信を中継･変換するためのデバイスやソフトウェアのことを指す． 簡単に言えば，異なる種類のネットワークやプロトコル間で通信を円滑に行うための出入り口のようなものである． 内部ネットワークと外部ネットワークの中継地である．

IX -2 -5 LAN

LAN（Local Area Network ）は， 狭い範囲内でコンピューターやデバイスがつながっているネットワークのことである． 学校内のコンピューター， 家庭内のデバイス， あるいはオフィス内のコンピューター同士が， ケーブルや無線などで直接つながっている状態を指す．

IX -2 -6 ファイアウォール

ファイアウォールは， 内部ネットワークを守るための防御壁のようなものである． 例えば，会社の警備員が不審な人が入ってこないように見張っているように， ネットワーク上においても不正アクセスや悪意あるプログラムから防衛する必要がある．

ファイアウォールは，ネットワークに入るデータや通信をチェックし， ウイルスや不正アクセスを遮断することで， コンピューターや情報を安全を確保する．

IX -2 -7 DMZ

DMZ は，DeMilitarized Zone（非武装地帯）の略であり， 外部と内部のネットワークの間においてファイアウォールで隔離された中立的なセグメント（区域）を指す． 外部のユーザーやデバイスがネットワークにアクセスする際の対話エリアであり， ここに置かれたサーバーやシステムが外部からのアクセスを受け付け，内部の重要なデータやシステムに直接アクセスされないようにする．

外部ネットワークにはあらゆる脅威が存在する． DMZ は， 内部への侵略を目論む攻撃者から内部情報資産を防衛するための仕組みとして機能する．

IX -3 基礎通信理論

IX -3 -1 プロトコル

IX -3 -2 パケット

X Hardware

X -1 PC の内容

X -1 -1 マザーボード

X -1 -2 CPU

X -1 -3 グラフィックボード

X -1 -4 GPU

X -1 -5 ROM

X -1 -6 RAM

XI Technical Writing

XII Typst

Bibliography

[1] 総務省， “インターネットの仕組み”． [Online]. Available at: https://www.soumu.go.jp/main_sosiki/cybersecurity/kokumin/basic/basic_service_02.html

APPENDIX A: Foo

APPENDIX B: Bar