【情報】 概論

目次

基礎理論

■データの単位

- ▶ ビット (bit)
- ▶ バイト (byte)
- ▶ ※ 32bit (=4byte) は 2^{32} ≒ 43億 通りの状態を表せ、8桁の16進数で表現できる。
  64bit (=8byte) は 2^{64} ≒ 1845京 通りの状態を表せ、16桁の16進数で表現できる。

アルゴリズムとプログラミング

■プログラミング

- ▶ トークン
- ▶ エスケープ文字
- ▶ スコープ
- ▶ シバン
- ▶ 丸め誤差
- ▶ シャドーイング
- ▶ 大域脱出

【情報】 概論

目次

- 基礎理論
- アルゴリズムとプログラミング
- ハードウェアとコンピュータ構成要素
- ソフトウェア
- データベース
- ネットワーク
- 情報セキュリティ
- ソフトウェア設計

基礎理論

■データの単位

- ▶ ビット (bit)　データの最小単位。`0` か `1` のいずれかの値。
- ▶ バイト (byte)　8bitの集まり。`00000000` ～ `11111111` の2^8=256通りの状態を表せる。
  16進数では2桁で表現できる。
- ▶ ※ 32bit (=4byte) は 2^{32} ≒ 43億 通りの状態を表せ、8桁の16進数で表現できる。
  64bit (=8byte) は 2^{64} ≒ 1845京 通りの状態を表せ、16桁の16進数で表現できる。

アルゴリズムとプログラミング

■プログラミング

- ▶ トークン　ソースコード上の文字列の最小単位
- ▶ エスケープ文字　それに続く文字について別の解釈をすることを示す文字　※ `\` 等
- ▶ スコープ　変数の有効範囲
- ▶ シバン　インタプリタを指定するためにスクリプト冒頭に書く行
- ▶ 丸め誤差　ある桁以降の値を捨ててしまうことにより生じる誤差
- ▶ シャドーイング　既存のものと同名の変数や関数を定義して、
  そのスコープで既存の変数や関数にアクセスできなくする機能
- ▶ 大域脱出　幾重の反復や関数呼び出しを全て終了して所定の場所へ飛ぶ

▶ 同期・非同期	実行結果を待つかどうか
▶ 糖衣構文	冗長なコードに代わる簡潔な書き方
▶ リファクタリング	プログラムの外部から見た動作を変えずに ソースコードの内部構造を整理すること
▶ ボイラープレート コード	ほとんどまたは全く変化することなく複数の場所で繰り返される 定型コードのセクション
▶ スニペット (コードスニペット)	様々なプログラムに挿入して利用することができる、 特定の機能を実現した短いコードのまとまり
▶ TODOコメント	あとでやりたい、やるべき作業に備忘録としてつけるコメント
▶ オーダ	入力データの大きさにたいするアルゴリズムの実行時間の増加量
▶ バイナリファイル	特定の文字コードの範囲に収まらない任意のビット列を含む バイナリデータを保存したファイル。テキストファイル以外の ファイルはすべてバイナリファイル。
▶ プロセッサ	CPUのこと
▶ マシン語	プロセッサが直接実行できる、0 と 1 の並びで書かれた言語
▶ アセンブリ言語	人間にわかりやすくした言語。マシン語と1対1対応している
▶ 高級言語	人間にさらにわかりやすくした言語。読み書きもしやすい
▶ コンパイル言語	ソースコードを事前にコンパイルして、 先にマシン語に変換しておく必要がある高級言語
▶ スクリプト言語	ソースコードを逐次マシン語に翻訳しつつ実行される高級言語
▶ コンパイラ	高級言語－(一気に)→アセンブリ言語やマシン語　にするソフト
▶ アセンブラ	アセンブリ言語→マシン語　にするソフト
▶ インタプリタ	高級言語－(1行ずつ)→マシン語　にするソフト
▶ コマンドライン	文字でコマンドを打ち込んで操作するインターフェイス
▶ シェル	人間がOSを操作するためのインターフェイス
▶ ターミナル	GUIの上でCUIの操作をしたいときに使用するアプリ
▶ コンソール	コマンド入力を受け付ける（一般には）黒い画面。
▶ 統合開発環境 (IDE)	エディタ・インタプリタ・デバッガなどを 1 画面でできるソフト
▶ ORM	RDBに対するデータの操作をオブ指向で扱えるようにする手法
▶ フレームワーク	Webアプリ・システム開発に必要な機能が予め用意された枠組み
▶ DSL (ドメイン固有言語)	何か特別な目的を実現するために定義された、人間にとっても 機械にとっても読みやすいテキストファイルの記述ルール

▶ ※ メモ用のコメントとして `# notes:` は便利そう。

▶ スロットリング

■コーディング（コードを書く行為）

▶ メタ構文変数

▶ メソッドチェーン

▶ モンキーパッチ

▶ オーバーライド

▶ ※ アーリーリターンでコードをスッキリさせようとするも、結局どの分岐にも当てはまらないためにスルーさせてしまうという間違いを犯すことがあるので注意。

▶ スパゲッティコード

■型システム

▶ 静的型付け
動的型付け

▶ 強い型付け
弱い型付け

▶ 明示的型変換
暗黙的型変換

- 名前的型システム
構造的型システム

▶ ダックタイピング

▶ ポリモーフィズム

■命名規則

▶ キャメルケース、ローワーキャメルケース

▶ パスカルケース、アッパーキャメルケース

▶ スネークケース

▶ アッパースネークケース、コンスタントケース

▶ ケバブケース、チェインケース

■オブジェクト指向

▶ クラス

▶ オブジェクト（ｲﾝｽﾀﾝｽ）

▶ ※ メモ用のコメントとして `# notes:` は便利そう。

▶ スロットリング　プログラムがアイテムを処理する速度をあえて制限すること

■コーディング（コードを書く行為）

▶ メタ構文変数　hoge など

▶ メソッドチェーン　変数を介さず、関数の返り値に直接関数を呼び出すスタイル

▶ モンキーパッチ　既存のクラスの**メソッド**の振る舞いを変更すること

▶ オーバーライド　・継承した親クラスのメソッドを上書きする
・返り値の型や仮引数が異なる関数をつくる

▶ ※ アーリーリターンでコードをスッキリさせようとするも、結局どの分岐にも当てはまらないためにスルーさせてしまうという間違いを犯すことがあるので注意。

▶ スパゲッティコード　処理の流れや構造が把握しにくく見通しの悪いソースコード

■型システム

▶ 静的型付け　値やオブの型安全性を、コンパイル時に検証する
動的型付け　〃　実行時に　〃

▶ 強い型付け　データ値の型を、プログラミング言語仕様内で解釈する
弱い型付け　言語仕様外の明示的型変換と型キャストを用いて様々に解釈できる

▶ 明示的型変換　変数を宣言する際、その型を人間が明示的に識別する必要がある機構
暗黙的型変換　周辺情報および文脈などから自動的に各々の型を決定する機構

- 名前的型システム
構造的型システム

▶ ダックタイピング　オブのクラスが何であれそのメソが呼べれば良しとするスタイル

▶ ポリモーフィズム　ある1つの関数の呼び出しに対し、オブ毎に異なる動作をする

■命名規則

▶ キャメルケース、ローワーキャメルケース　hogeHogeHoge

▶ パスカルケース、アッパーキャメルケース　HogeHogeHoge

▶ スネークケース　hoge_hoge_hoge

▶ アッパースネークケース、コンスタントケース　HOGE_HOGE_HOGE

▶ ケバブケース、チェインケース　hoge-hoge-hoge

■オブジェクト指向

▶ クラス　オブジェクトの設計図（ひな形）

▶ オブジェクト（ｲﾝｽﾀﾝｽ）　クラスをもとに作られたデータの集合体

- ▶ メソッド
- ▶ レシーバ
- ▶ メッセージ
- ▶ 状態（ｽﾃｰﾄ）
- ▶ 属性（ｱﾄﾘﾋﾞｭｰﾄ、ﾌﾟﾛﾊﾟﾃｨ）

- ▶ メンバ変数（ｲﾝｽﾀﾝｽ変数）
- ▶ ゲッターメソッド
- ▶ セッターメソッド
- ▶ アクセサメソッド

■文字コード

- ▶ ※ `Shift-JIS` と `CP932` (= `MS932` = `Windows-31J`) はとてもよく似ている。
- ▶ ☆ 改行コードについて

■正規表現

- ▶ ☆ 正規表現の規格（参考）
- ▶ ☆ マッチする文字列を視覚的に確認するためのサイト

メタ文字

- ▶ 任意の1文字
- ▶ 文字群のいずれか1文字
- ▶ 文字群のいずれでもない1文字
- ▶ 数字1文字・英数字か_の1文字
- ▶ 数字以外・英数字と_以外の1文字
- ▶ 行の先頭
- ▶ 行の末尾
- ▶ 直前のパターンを0回か1回繰り返す
- ▶ 直前のパターンを丁度*n*回くり返す
- ▶ 直前のパターンを0回以上くり返す
- ▶ 直前のパターンを1回以上くり返す
- ▶ 直前のパターンを*n*回以上くり返す
- ▶ 直前のパターンを*m*回以下くり返す

- ▶ メソッド　オブジェクトがもつ動作や振る舞い
- ▶ レシーバ　とくにメソッドと比較したときのオブジェクトのこと
- ▶ メッセージ　とくにレシーバと比較したときのメソッドのこと
- ▶ 状態（ｽﾃｰﾄ）　オブジェクトごとに保持されるデータのこと
- ▶ 属性（ｱﾄﾘﾋﾞｭｰﾄ、ﾌﾟﾛﾊﾟﾃｨ）　オブジェクトから取得、あるいはそれに設定できる値

- ▶ メンバ変数（ｲﾝｽﾀﾝｽ変数）　個々のインスタンスごとに固有の変数
- ▶ ゲッターメソッド　メンバ変数を読み取るメソッド
- ▶ セッターメソッド　メンバ変数を書き換えるメソッド
- ▶ アクセサメソッド　ゲッターメソッド、セッターメソッドを併せてこう呼ぶ

■文字コード

- ▶ ※ `Shift-JIS` と `CP932` (= `MS932` = `Windows-31J`) はとてもよく似ている。
- ▶ ☆ 改行コードについて

■正規表現

- ▶ ☆ 正規表現の規格（参考）
- ▶ ☆ マッチする文字列を視覚的に確認するためのサイト

メタ文字

- ▶ 任意の1文字　(.|\n)　※ `.` だけだと `\n` が含まれない
- ▶ 文字群のいずれか1文字　[*characters*]　[*charA-charZ*]
- ▶ 文字群のいずれでもない1文字　[^*charcters*]
- ▶ 数字1文字・英数字か_の1文字　\d　・　\w
- ▶ 数字以外・英数字と_以外の1文字　\D　・　\W
- ▶ 行の先頭　^
- ▶ 行の末尾　$
- ▶ 直前のパターンを0回か1回繰り返す　?
- ▶ 直前のパターンを丁度*n*回くり返す　{*n*}
- ▶ 直前のパターンを0回以上くり返す　*（欲張り）　*?（控えめ）
- ▶ 直前のパターンを1回以上くり返す　+（欲張り）　+?（控えめ）
- ▶ 直前のパターンを*n*回以上くり返す　{*n*,}（欲張り）　{*n*,}?（控えめ）
- ▶ 直前のパターンを*m*回以下くり返す　{,*m*}（欲張り）　{,*m*}?（控えめ）

- ▶ 直前のパターンを*n*〜*m*回くり返す
- ▶ 囲まれたパターンをグループ化
- ▶ 区切られたパターンのいずれか
- ▶ キャプチャで抜き出したい部分
- ▶ キャプチャを回避してグループ化
- ▶ 名前をつけてキャプチャ
- ▶ *next*が続いている*prev*
- ▶ *next*が続いていない*prev*
- ▶ *prev*が前にある*next*
- ▶ *prev*が前にない*next*
- ▶ メタ文字をエスケープ
- ▶ ☆ 主なメタ文字のまとめ（具体例つき）

■文化

- ▶ ハッカソン
- ▶ イースターエッグ

- ▶ 直前のパターンを*n*〜*m*回くり返す　{*n*,*m*}（欲張り）　{*n*,*m*}?（控えめ）
- ▶ 囲まれたパターンをグループ化　(*pattern*)
- ▶ 区切られたパターンのいずれか　|
- ▶ キャプチャで抜き出したい部分　(*pattern*)
- ▶ キャプチャを回避してグループ化　(?:*pattern*)
- ▶ 名前をつけてキャプチャ　(?<*name*>*pattern*)
- ▶ *next*が続いている*prev*　*prev*(?=*next*)
- ▶ *next*が続いていない*prev*　*prev*(?!*next*)
- ▶ *prev*が前にある*next*　(?<=*prev*)*next*
- ▶ *prev*が前にない*next*　(?<!*prev*)*next*
- ▶ メタ文字をエスケープ　\　※エスケープしたい文字の直前におく
- ▶ ☆ 主なメタ文字のまとめ（具体例つき）

■文化

- ▶ ハッカソン　ソフト開発関係者らが集まって集中的にソフト開発作業を行うイベント
- ▶ イースターエッグ　ソフトなどで密かに備えられている、本来の機能、目的とは無関係のメッセージや画面

ハードウェアとコンピュータ構成要素

■CPU

- ▶ ビット数
- ▶ オーバーヘッド

■ディスク

- ▶ デフラグ

ハードウェアとコンピュータ構成要素

■CPU

- ▶ ビット数　1バイトのデータを1度に扱える量。64bitなら約1845京個を扱える
- ▶ オーバーヘッド　ある処理を行うために必要となる付加的、間接的な処理や手続き。また、そのために機器やシステムへかかる負荷、余分に費やされる処理時間

■ディスク

- ▶ デフラグ　ディスク上のファイルの断片化を解消すること

ソフトウェア

■タスク（プロセス）管理

ソフトウェア

■タスク（プロセス）管理

- マルチタスク
- プリエンプション
- **非**協調的（プリエンプティブ）マルチタスク
- 協調的（**ノン**プリエンプティブ）マルチタスク

■プログラム・ソフトウェア

- メジャー・バージョン番号
- マイナー・バージョン番号
- ミドルウェア
- ☆ インストーラの名前にある x86_64, amd64, i386, i686, x86, x64 の意味
- キック
- プロファイリング

■サーバソフトウェア

- バーチャルホスト
- Webサーバ
- APサーバ
- DBサーバ

■Web用ツール

- 静的サイトジェネレータ（SSG）
- コンテンツ管理システム（CMS）

データベース

■一般

- ※ データベースに使われる拡張子は、圧倒的に `.db` が多い。

■データモデル

- マルチタスク　システムなどが同時に複数のタスクを並行して実行すること
- プリエンプション　プログラム側の挙動に頼らずOS側から実行を中断させられる機能
- **非**協調的（プリエンプティブ）マルチタスク　プリエンプションを用いた制御方式によるマルチタスク。OSがCPU割当てを管理するため強制的にタスク切替えができる。現在のコンピュータではこの方式のマルチタスクが主流。
- 協調的（**ノン**プリエンプティブ）マルチタスク　プリエンプションを用いない制御方式のマルチタスク。実行中のタスクは中断できず、それが自発的に制御権をOSに返し、OSが他のタスクにまた制御を渡すことで、タスクが切り替わる。

■プログラム・ソフトウェア

- メジャー・バージョン番号　Python 3.7.1 の 3 の部分
- マイナー・バージョン番号　Python 3.7.1 の 7.1 の部分
- ミドルウェア　OSとアプリケーションの間に入り、両者の役割を補佐するソフト
- ☆ インストーラの名前にある x86_64, amd64, i386, i686, x86, x64 の意味
- キック　プログラムからスリープ状態の別のプログラムを起動すること
- プロファイリング　プログラムが実行されるようすを監視・記録し、プログラムの各箇所の動作順や実行時間などを集計・解析すること

■サーバソフトウェア

- バーチャルホスト　1台のサーバで仮想的に複数のドメインを運用するサーバ技術。
- Webサーバ　ブラウザから検索した検索結果を視覚的に表示させる役割を担うサーバ
- APサーバ　動的コンテンツや業務処理を行うサーバ
- DBサーバ　DBMSが動作しているサーバ

■Web用ツール

- 静的サイトジェネレータ（SSG）　未加工データとテンプレート群をもとに、完全に静的なHTML Webサイトを生成するツール
- コンテンツ管理システム（CMS）　専門的な知識なしに、Webサイト用のテキストや画像を作成、編集、整理、公開できるツール（例: WordPress）

データベース

■一般

- ※ データベースに使われる拡張子は、圧倒的に `.db` が多い。

■データモデル

▶ ※ データモデルとは、現実世界にある複雑なデータの相互関係をわかりやすく表した図。**データベースを作る際の設計図**になる。

- 関係データモデル：データの集まりを「表」で表す。関係モデルともいう。
- 階層データモデル
- ネットワークデータモデル

■関係データベースと関係データモデルでの用語の違い

- テーブル（表）　⇄　関係
- レコード　⇄　組
- 列　⇄　属性

■正規化

▶ 正規化
▶ 正規化の目的
▶ 関数従属
▶ 完全関数従属
▶ 部分関数従属
▶ 推移的関数従属
▶ 第１正規化
▶ 第２正規化
▶ 第３正規化

■データベース管理システム（DBMS）

▶ ※ DBMSは、アプリケーションソフトウェアの要求に応じてデータベースを操作するシステム。
▶ DBMSの3機能
▶ ストアドプロシージャ
▶ RDBMS
▶ SQL
▶ NoSQL

データ操作

▶ 選択
▶ 射影

▶ ※ データモデルとは、現実世界にある複雑なデータの相互関係をわかりやすく表した図。**データベースを作る際の設計図**になる。

- 関係データモデル：データの集まりを「表」で表す。関係モデルともいう。
- 階層データモデル
- ネットワークデータモデル

■関係データベースと関係データモデルでの用語の違い

- テーブル（表）　⇄　関係
- レコード　⇄　組
- 列　⇄　属性

■正規化

▶ 正規化　複数の表を用意することでデータ管理を合理化すること
▶ 正規化の目的　表中のすべての関数従属を、完全関数従属だけにする
▶ 関数従属　ある属性の値が決まると他の属性の値も一意に決まること
▶ 完全関数従属　属性が主キー（複合主キーふくむ）に関数従属
▶ 部分関数従属　属性が複合主キーの一部に関数従属
▶ 推移的関数従属　属性が主キー（複合主キーふくむ）以外の属性に関数従属
▶ 第１正規化　属性の繰り返しがない状態にする。レコードを追加して実現。
▶ 第２正規化　部分関数従属がない状態にすること。テーブルを分割して実現。
▶ 第３正規化　推移的関数従属がない状態にすること。テーブルを分割して実現。

■データベース管理システム（DBMS）

▶ ※ DBMSは、アプリケーションソフトウェアの要求に応じてデータベースを操作するシステム。

▶ DBMSの3機能　データ操作、トランザクション管理、排他制御
▶ ストアドプロシージャ　複数の命令を１つにまとめてDBMSに保存したもの
▶ RDBMS　**関係データベース**管理システム
▶ SQL　**関係データベース**を操作するための言語
▶ NoSQL　RDBMS以外のDBMSのこと

データ操作

▶ 選択　テーブルからある特定のレコードのみを取り出す操作
▶ 射影　テーブルからある特定の列のみを取り出す操作

- ▶ 結合
- ▶ 挿入
- ▶ 更新
- ▶ 削除

結合

- ▶ 内部結合
- ▶ 左外部結合
- ▶ 右外部結合
- ▶ 完全外部結合
- ▶ 交差結合

トランザクション管理

- ▶ トランザクション
- ▶ ACID特性
- ▶ Atomicity
- ▶ Consistency
- ▶ Isolation
- ▶ Durability
- ▶ 障害回復（リカバリ）
- ▶ バックアップファイル
- ▶ バックアップ
- ▶ 更新前・後ログファイル
- ▶ ロールバック
- ▶ ロールフォワード

排他制御

- ▶ 排他制御
- ▶ ロック
- ▶ 専有ロック
- ▶ 共有ロック

- ▶ 結合　複数のテーブルを１つにする操作
- ▶ 挿入　テーブルにレコードを追加する操作
- ▶ 更新　レコード内のデータを変更する操作
- ▶ 削除　テーブルからレコードを削除する操作

結合

- ▶ 内部結合　2つの表の合体可能な録のみ取り出す
- ▶ 左外部結合　表1の全録を取り出して、それに表2の録をくっつける
- ▶ 右外部結合　表2の全録を取り出して、それに表1の録をくっつける
- ▶ 完全外部結合　両表の全録を取り出して、くっつけられる範囲でくっつける
- ▶ 交差結合　両表の全録を取り出して、すべての組み合わせでくっつける

トランザクション管理

- ▶ トランザクション　途中で終わってはならない、関連する複数の処理のまとまり
- ▶ ACID特性　原子性 Atomicity, 一貫性 Consistency, 独立性 Isolation, 耐久性 Durability
- ▶ Atomicity　「全く処理していない」か「完了」かのどちらかの結果になること
- ▶ Consistency　トランザクション前後でDBに矛盾がないこと
- ▶ Isolation　単独処理でも並行処理でも同じ結果になること
- ▶ Durability　正常終了したら、そのあと障害があっても結果が変わらないこと
- ▶ 障害回復（リカバリ）　壊れたデータやハードウェアを直すこと
- ▶ バックアップファイル　データベース全体を保存しているファ
- ▶ バックアップ　バックアップファイルを取得する処理
- ▶ 更新前・後ログファイル　トランザクション開始直前・完了直後の状態を保存したファ
- ▶ ロールバック　トランザクション開始直前の状態に戻す障害回復手法
- ▶ ロールフォワード　トランザクション完了直後の状態に戻す障害回復手法

排他制御

- ▶ 排他制御　複数の人が同じデータを同時に更新しようとした場合に
データに矛盾が生じないようにする機能
- ▶ ロック　データベースに対するデータの読み書きを一時的に制限する機能
- ▶ 専有ロック　ロックをかけたトランザクションのみがデータを読み書きできる
- ▶ 共有ロック　ロックをかけたトランザクションと他のそれの両方がデータを読める

- ロックの両立性
- ロックの粒度

■DDLにおける制約の種類

- 非NULL制約
- 一意性制約
- 主キー制約
- 外部キー制約（参照制約）
- 検査制約

■その他の用語

- トリガー
- インデックス

- ロックの両立性　複数のトランザクションが共有ロックを同時にかけられる特性
- ロックの粒度　ロックの対象となるデータの単位（DB単位、表単位、録単位、...）

■DDLにおける制約の種類

- 非NULL制約　列の値がNULLであることを許可しない
- 一意性制約　列の値が重複する行を作成できない
- 主キー制約　表のなかで行を一位に特定できる主キーの列を指定する
- 外部キー制約（参照制約）　外部キー（別の表の主キーを取り込んだ列）として指定した列の値と同じ値を持つ主キーの列が参照先の表に存在しなければならない
- 検査制約　検査条件を満たさない値を格納することはできない

■その他の用語

- トリガー　表の更新（録の追加、変更、削除）の前後に、自動で他の表も更新
- インデックス　ある列を予めグループ化させておくことで検索を高速化を図る仕組み

ネットワーク

■回線

- 回線
- 回線速度
- 伝送速度
- 伝送効率
- 伝送時間

■伝送方式

- 回線交換方式
- パケット交換方式
- パケット
- ヘッダ

■通信

- ICT
- ホスティングサービス
- 輻輳(ふくそう)

ネットワーク

■回線

- 回線　データが通る線。伝送路、通信回線とも。
- 回線速度　その回線でデータをやり取りできる最大の速さ [bps]
- 伝送速度　その回線でデータをやり取りする実際の速さ [bps]
- 伝送効率　伝送速度 ÷ 回線速度
- 伝送時間　あるデータ量 ÷ 伝送速度　※転送時間とも。

■伝送方式

- 回線交換方式　回線を占有して情報をやり取り
- パケット交換方式　データをパケット単位に分割し、複数の回線で情報をやり取り
- パケット　データを小分けにし、それぞれにヘッダをつけた単位
- ヘッダ　パケットの宛先などの説明書き。制御情報とも。

■通信

- ICT　情報通信技術（Information and Communication Technology）
- ホスティングサービス　データセンター内に設置されたサーバを貸し出すサービス
- 輻輳(ふくそう)　インターネット回線や電話回線にアクセスが集中すること

▶ スループット

▶ ☆ アクセスポイント

▶ ☆ HTTPステータスコード

■プロトコル

▶ イーサネット

■ウェルノウンポート

▶ ※ ポート番号は `0` ～ `65535` まで存在するが、そのうち `0` ～ `1023` はよく利用されるアプリケーション用のポート番号としてみなされており、IANAがウェルノウンポートとして管理している。

- TCP 20番　FTP（データ）
- TCP 21番　FTP（制御）
- TCP 22番　SSH
- TCP 23番　Telnet
- TCP 25番　SMTP
- UDP 53番　DNS
- UDP 67番　DHCP（サーバ）
- UDP 68番　DHCP（クライアント）
- TCP 80番　HTTP
- TCP 110番　POP3
- UDP 123番　NTP　(=Network Time Protocol)
- TCP 143番　IMAP4
- UDP 161番　SNMP
- UDP 162番　SNMP Trap
- TCP 443番　HTTPS

▶ ※ ちなみに、`1024` ～ `49151` も特定のアプリケーションなどが使用するポートとして、IANAがレジスタードポート番号と呼んで管理している。

■ネットワーク

▶ ※ IPアドレスは前方の「ネットワーク部」と後方の「ホスト部」で構成される。

▶ サブネットマスク

▶ スループット

▶ ☆ アクセスポイント

▶ ☆ HTTPステータスコード

■プロトコル

▶ イーサネット　有線LANに関する最も支配的な規格。
また、その規格を満たしたケーブルや端子のことも指す。

■ウェルノウンポート

▶ ※ ポート番号は `0` ～ `65535` まで存在するが、そのうち `0` ～ `1023` はよく利用されるアプリケーション用のポート番号としてみなされており、IANAがウェルノウンポートとして管理している。

- TCP 20番　FTP（データ）
- TCP 21番　FTP（制御）
- TCP 22番　SSH
- TCP 23番　Telnet
- TCP 25番　SMTP
- UDP 53番　DNS
- UDP 67番　DHCP（サーバ）
- UDP 68番　DHCP（クライアント）
- TCP 80番　HTTP
- TCP 110番　POP3
- UDP 123番　NTP　(=Network Time Protocol)
- TCP 143番　IMAP4
- UDP 161番　SNMP
- UDP 162番　SNMP Trap
- TCP 443番　HTTPS

▶ ※ ちなみに、`1024` ～ `49151` も特定のアプリケーションなどが使用するポートとして、IANAがレジスタードポート番号と呼んで管理している。

■ネットワーク

▶ ※ IPアドレスは前方の「ネットワーク部」と後方の「ホスト部」で構成される。

▶ サブネットマスク　ネットワーク部とホスト部を区切るための番号。2進数のIPアドレスにてネットワーク部をすべて1に、ホスト部をすべて0に置換したもの。

- プレフィックス長（サブネットマスク長）
- (アドレス) プレフィックス表記法 (**CIDR表記**
- **サブネット**
- **※ サブネット外へのアクセスにはルーティングが必要となる。そのため、ルーティングテーブルやファイアウォールによるアクセス制御は基本的にサブネット単位で行われる。ホストのグルーピングやセキュリティ向上のためにサブネットは重要な要素となる。**
- **ネットワークアドレス**
- **ブロードキャストアドレス**
- **※ プライベートIPアドレス（IPv4）の範囲は、`10.0.0.0` ～ `10.255.255.255` 、`172.16.0.0` ～ `172.31.255.255` 、`192.168.0.0` ～ `192.168.255.255` 。**
- **※ `127.0.0.1` は自分自身を指すIPアドレスで、ローカルループバックアドレスといわれる。`localhost` でも参照できる（できないこともある）。**
- **DHCP**
- **DNS**
- **トップレベルドメイン**
- **ホスト名とは**
- **ドメインに対応するIPアドレスを調べる**
- **ﾈｯﾄﾜｰｸ上のパケットの経路を調べる**
- **パケット・パケット通信とは**
- **URL**
- **URN**
- **URI**
- **URLにおけるスキーム**
- **URLにおけるオーソリティ**
- **URLにおけるパス**
- **URLにおけるクエリ**
- **URLにおけるフラグメント**
- **ルーティング**

- プレフィックス長（サブネットマスク長）　ネットワーク部の桁数
- (アドレス) プレフィックス表記法 (**CIDR表記**)　IPアドレス末尾にスラッシュで区切ってプレフィックス長を書く表記法（例：`172.31.1.0/24` ）
- サブネット　ネットワーク部が等しいIPアドレス範囲
- ※ **サブネット外へのアクセスにはルーティングが必要**となる。そのため、ルーティングテーブルやファイアウォールによるアクセス制御は基本的にサブネット単位で行われる。**ホストのグルーピング**や**セキュリティ向上**のためにサブネットは重要な要素となる。
- ネットワークアドレス　2進数のIPアドレスにて、ホスト部をすべて0にしたもの。サブネットを識別するための番号として使われる
- ブロードキャストアドレス　2進数のIPアドレスにて、ホスト部をすべて1にしたもの。同一ネットワーク内の全ホストにデータを送る際に使われる
- ※ プライベートIPアドレス（IPv4）の範囲は、`10.0.0.0` ～ `10.255.255.255` 、`172.16.0.0` ～ `172.31.255.255` 、`192.168.0.0` ～ `192.168.255.255` 。
- ※ `127.0.0.1` は自分自身を指すIPアドレスで、ローカルループバックアドレスといわれる。`localhost` でも参照できる（できないこともある）。
- DHCP　IPアドレスを各端末に自動的に割り当てるプロトコル
- DNS　IPアドレスとドメインを対応させるシステム
- トップレベルドメイン　ドメイン名にてドットで区切られた文字列の一番右の部分
- ホスト名とは　ネットワークに接続された機器（ホスト）に付けられた名前
- ドメインに対応するIPアドレスを調べる　cmdにて `> nslookup ドメイン` と打つ
- ﾈｯﾄﾜｰｸ上のパケットの経路を調べる　cmdにて `> tracert ドメイン` と打つ
- パケット・パケット通信とは　通信上、分割されたデータの単位
- URL　リソースのネット上での所在を表す書式
- URN　リソースを**その存在位置によらずに**一意に識別するための符号を与える書式
- URI　URL と URN の総称
- URLにおけるスキーム　http://example.com/main/index?q=python#lead なら http
- URLにおけるオーソリティ　〃 なら example.com
- URLにおけるパス　〃 なら /main/index
- URLにおけるクエリ　〃 なら q=python
- URLにおけるフラグメント　〃 の lead
- ルーティング　ネットワーク上でデータを転送する際、その経路を導き出すこと

- ▶ **ポート転送**
- ▶ **IPAM**

■インターネット

- ▶ **検索エンジン**
- ▶ **SEO**
- ▶ **☆ Webサイトの開発において、サーバにアップロードしたもののページが更新されない場合、閲覧用のブラウザにキャッシュが残っている可能性があるので、スーパーリロードしよう。**

■サーバ

- ▶ **専用サーバ**
- ▶ **VPS**
  (仮想専用サーバ)
- ▶ **共用サーバ**

- ▶ ポート転送　ある機器のポートへの通信を別機器のポートへ転送する機能
- ▶ IPAM　ネットワーク内で運用するIPアドレスの一元的管理（用のソフト）

■インターネット

- ▶ 検索エンジン　ネット上にある情報を検索する機能、およびそのプログラム
- ▶ SEO　検索エンジンの検索結果上位にサイトが表示されるための様々な工夫
- ▶ ☆ Webサイトの開発において、サーバにアップロードしたもののページが更新されない場合、閲覧用のブラウザにキャッシュが残っている可能性があるので、スーパーリロードしよう。

■サーバ

- ▶ 専用サーバ　1台の物理サーバを丸ごと借り受けて専有する形式のサーバ
- ▶ VPS
  (仮想専用サーバ)　仮想化技術で1台の物理サーバ上に構築された複数台の仮想サーバのひとつを借り受けて専有する形式のサーバ
- ▶ 共用サーバ　1台の物理サーバーを複数人で借り受けて専有する形式のサーバ

情報セキュリティ

■人的脅威

- ▶ **ソーシャル**
  エンジニアリング
- ▶ **不正の**
  トライアングル

■技術的脅威

攻撃の準備

- ▶ **ポートスキャン**
- ▶ **セキュリティホール**

パスワードを割り出す攻撃

- ▶ **ブルートフォース攻撃**
- ▶ **辞書攻撃**
- ▶ **パスワードリスト攻撃**

マルウェアによる攻撃

情報セキュリティ

■人的脅威

- ▶ ソーシャル
  エンジニアリング　特別なツールや技術を使わず、人間の心理的な隙を利用して機密情報を手に入れること
- ▶ 不正の
  トライアングル　人が不正を働くのは、機会・動機・正当化の３条件が揃った時に限られるという理論

■技術的脅威

攻撃の準備

- ▶ ポートスキャン　攻撃できそうなサービスがあるかどうかの事前調査
- ▶ セキュリティホール　システムに存在する欠陥。脆弱性。

パスワードを割り出す攻撃

- ▶ ブルートフォース攻撃　可能な文字列を総当たりで試して割り出す
- ▶ 辞書攻撃　辞書や人名録などに載っている全単語を使って割り出す
- ▶ パスワードリスト攻撃　ほかのサービスから不正に入手したID・PWの一覧を使う

マルウェアによる攻撃

- **※ マルウェア（Malware）とは、悪意のあるソフトウェアの総称である。**
- **ボット**
- **スパイウェア**
- **ランサムウェア**
- **キーロガー**
- **バックドア**

Webサイトに仕掛けられる攻撃

- **フィッシング**
- **ドライブバイダウンロード**
- **SEOポイズニング**
- **DNSキャッシュポイズニング**
- **SQLインジェクション**
- **クロスサイト スクリプティング**
- **☆ クロスサイトリクエストフォージェリ（CSRF）**

その他の攻撃

- **スパム**
- **DoS攻撃**
- **DDoS攻撃**
- **ディレクトリトラバーサル攻撃**

■物理的攻撃

- **※ 物理的攻撃は、ネットワークなどを使わない直接的な手段によって引き起こされる脅威。**
- **具体例は**

ソフトウェア設計

■UML

- ※ マルウェア（Malware）とは、悪意のあるソフトウェアの総称である。
- ボット　攻撃者から遠隔で指令を受けて動作するプログラム
- スパイウェア　利用者に気づかれないように個人情報などを収集するプログラム
- ランサムウェア　身代金を要求するプログラム
- キーロガー　キーボード入力を記録するプログラム
- バックドア　システムに不正アクセスするための裏口。

Webサイトに仕掛けられる攻撃

- フィッシング　利用者を偽サイトに誘導し、個人情報を入力させる
- ドライブバイダウンロード　サイトの閲覧だけでマルウェアをダウンロードさせる
- SEOポイズニング　SEOを行って悪質サイトを検索結果上位に表示させる
- DNSキャッシュポイズニング　DNSサーバにキャッシュされているドメイン名とIPアドレスの対応を置き換えることで悪質サイトに誘導
- SQLインジェクション　SQLの実行によるデータベース改竄、不正なデータ取得
- クロスサイト スクリプティング　ユーザの入力データが後で表示されるようなフォームに第三者が悪意あるスクリプトを書くことでデータを盗む
- ☆ クロスサイトリクエストフォージェリ（CSRF）

その他の攻撃

- スパム　受信者の承諾なしに無差別に送付されるメール
- DoS攻撃　メールやリクエストを大量に送り、サービスを提供不能に
- DDoS攻撃　多数のコンピュータによるDoS攻撃
- ディレクトリトラバーサル攻撃　管理者の意図しないファイルを不正に閲覧

■物理的攻撃

- ※ 物理的攻撃は、ネットワークなどを使わない直接的な手段によって引き起こされる脅威。
- 具体例は　台水害、落雷、地震、大気汚染、爆発、火災、侵入、盗難など

ソフトウェア設計

■UML

▶ **※ UML (Unified Modeling Language、統一モデリング言語) とは、システムの振る舞いや構造をオブジェクト指向で分析したり設計したりする際、図を用いることで視覚的に把握できるようになり、効果的に表現できる。しかし、その図の描き方が人によって違っては困るために、標準規格として作られた図の記法（モデリング言語）がUMLである。**

構造図

- ▶ **クラス図**
- ▶ **コンポジット構造図（複合構造図）**
- ▶ **コンポーネント図**
- ▶ **オブジェクト図**
- ▶ **配置図**
- ▶ **パッケージ図**

振舞い図

- ▶ **アクティビティ図**
- ▶ **ユースケース図**
- ▶ **ステートマシン図（状態遷移図）**
- ▶ **シーケンス図**
- ▶ **コミュニケーション図**
- ▶ **インタラクションオーバービュー図（相互作用概要図）**
- ▶ **タイミング図**

■その他の設計図

- ▶ **E-R図**
- ▶ **DFD (データフロー図)**

■ソフトウェア設計

- ▶ **MVC**
- ▶ **CRUD**
- ▶ **リグレッション**

▶ ※ UML (Unified Modeling Language、統一モデリング言語) とは、システムの振る舞いや構造をオブジェクト指向で分析したり設計したりする際、図を用いることで視覚的に把握できるようになり、効果的に表現できる。しかし、その図の描き方が人によって違っては困るために、標準規格として作られた図の記法（モデリング言語）がUMLである。

構造図

- ▶ クラス図　クラス間の関係を表現
- ▶ コンポジット構造図（複合構造図）　複数のクラスを包括するようなクラスやコンポーネントにおいて、その内部要素の構造や相互作用を表現
- ▶ コンポーネント図　コンポーネントの内部構造やそれら同士の相互作用を表現
- ▶ オブジェクト図　オブジェクト（インスタンス）間の関係を表現
- ▶ 配置図　システムの物理的側面を表現
- ▶ パッケージ図　モデリングされたクラスなどをグループ化し関連付け

振舞い図

- ▶ アクティビティ図　ある振る舞いから次のふるまいへの制御の流れを表現
- ▶ ユースケース図　システムが外部に提供する機能と、それを利用する人や外部システムとの関係を表現
- ▶ ステートマシン図（状態遷移図）　1つのオブジェクトの状態がイベントの発生や時間の経過とともにどのように変化するかを表現
- ▶ シーケンス図　オブジェクト間の相互作用を時系列で表現
- ▶ コミュニケーション図　オブジェクト間の相互作用と応答を表現
- ▶ インタラクションオーバービュー図（相互作用概要図）　機能ごとに記述された相互作用図が、より広域のシステム構成から見たとき、それぞれがどのように連携しているのかを表現
- ▶ タイミング図　クラスやオブジェクトの状態を時系列で表現

■その他の設計図

- ▶ E-R図　データの関係性を実体、関連の2つの概念を用いて表現
- ▶ DFD (データフロー図)　データの流れを表現

■ソフトウェア設計

- ▶ MVC　機能を Model、View、Controller に分けて設計
- ▶ CRUD　永続的にデータを扱うソフトウェアに要求される4つの基本機能
- ▶ リグレッション　ある変更により混入された新たなバグ

■**ソフトウェア開発手法**

- **継続的インテグレーション（CI）**

システム設計

■**システム**

- **HA構成**

■**コンピューティング**

- **クラウドコンピューティング**
- **エッジコンピューティング**

■ソフトウェア開発手法

- 継続的インテグレーション（CI）　頻繁に（少なくとも毎日）コードをコミットして共有リポジトリにマージすれば、そのたびに自動でビルドとテストが実行されるような開発手法。品質を確保しつつ開発速度を向上させられる。

システム設計

■システム

- HA構成 (High Availability)　高可用性のシステム構成のこと。サーバやネットワークの物理的、論理的な冗長化によって災害や障害などでも継続稼働可能にする。

■コンピューティング

- クラウドコンピューティング　**全て**の情報をクラウドに集約しクラウド上の高性能サーバでデータを処理するコンピューティング手法。
- エッジコンピューティング　コンピュータネットワークの周縁部分でデータを処理し、加工した情報のみをクラウドに送信して配置する手法。