

# コンピュータ科学実験 1 指導書 ソフトウェア

名古屋大学情報学部

# まえがき

実験の目的としては以下のような事柄がある。

- (1) コンピュータ科学の基礎について，講義・演習などで得た知識を実験を通して検証・体得する。
- (2) コンピュータ科学の基礎となるシステムの構築に関して，実際に計算機ハードウェアおよびソフトウェアのシステムを構築することにより，その構成法を体得する。
- (3) 本実験においては，個人ワークとチームワークの二面性を持った課題が与えられている。このような課題の解決に向けての仕事の進め方を体得する。
- (4) 分からない問題に対して，文献・資料を調査したり他人に聞くなど工夫をこらし，自らがこれを解決する態度（自立的探求心と問題解決能力）を身につける。
- (5) 事象や現象を把握するための基本的な装置・道具（計測機器・ソフトウェア）について学び，その操作方法を会得する。
- (6) 実験レポートを記述する訓練を通じてレポートの記述形式を学ぶとともに，他人が理解できる文章を書く能力を習得する。

これらの事柄をふまえて，積極的に実験課題に取り込んで欲しい。

---

## レポート提出期限について

本実験では，大きく分けて2つの実験課題レポートを提出する。レポートの書き方についての指導を受けるため，また，実験スケジュールの目安として，各課題レポートの提出期限を下記の通りとする。

- 第2週終了の1週間後まで: 課題 1, 2, 3, 4, 5, 調査課題 1, 2, 3, 4, 5
- 第4週終了の1週間後まで: 課題 6, 調査課題 6

ただし，人間の記憶は時間とともに急速に減衰する。実験終了後は速やかに，各週分の実験結果をレポートにまとめておくことが望ましい。

# 実験の心得

## 1 実験に対しての心得

- (1) 実験中に人身事故などの災害が発生したときは、冷静かつ速やかに応急処置を行い、実験担当者に報告して指示を受けること。
- (2) 実験中にふざけたりしない。常に細心の注意を払い、人身事故や災害の無いように心がける。
- (3) 欠席や遅刻および無断退席はしない。なお、本人および親族等に緊急やむを得ない事由（病気・事故）が生じたときには、実験担当者に申し出て指示を受けること。無断欠席は不合格となる。
- (4) レポートの提出期限を厳守すること。
- (5) 指導書をあらかじめよく読み、実験の目的や実験方法・手順などを捉えておくこと。
- (6) グラフ用紙や筆記用具などを用意すること。
- (7) 予期しない結果があることがある。その場合は原因を追求・分析すること。
- (8) 難解な問題と思えるものは文献・資料などを調査して解決すること。
- (9) 実験が終了したとき、班の代表者は実験担当者に実験の進捗状況や実験装置・機器具の不具合、災害の有無等を報告すること。

## 2 実験レポートの書き方

- (1) 実験報告書は PDF ファイルとして作成する。
- (2) レポートの冒頭に氏名・学籍番号に加えて e-mail アドレスを記載する。レポート再提出などの連絡は、そのアドレスに送るのでメールを見落とさないこと。
- (3) レポートの作成には  $\text{\LaTeX}$  を使用することを強く推奨する。
  - 実験室の環境において使用できるコマンドがコンピュータ科学実験の Web ページ<sup>\*1</sup>に載っている。例えば以下のコマンドがある。  
 $\text{\LaTeX}$  ファイルのコンパイル: `platex main.tex`  
DVI ファイルの閲覧: `pxdvi main.dvi`  
DVI から PDF への変換: `dvipdfmx main.dvi`  
PDF ファイルの閲覧と印刷: `acroread main.pdf`  
DVI ファイルの印刷: `dvips -t a4 -q -f main.dvi | lpr`
  - その他の環境で使用する場合は、 $\text{\TeX}$  Wiki のサイト<sup>\*2</sup>を参照せよ。特に MS-Windows を使用する場合は、 $\text{\TeX}$  インストーラ 3 のサイト<sup>\*3</sup>が参考になる。
  - この他、 $\text{\LaTeX}$  を取り扱えるウェブサービスを使う方法も推奨される。以下は 2019 年始の推奨されるウェブサービスである。これらでは、実績のある (つまり古い)p $\text{\LaTeX}$  だけではなく、Lua $\text{\LaTeX}$  等が利用可能であり、またデフォルトになっているため注意が必要である。

---

<sup>\*1</sup> <http://www.ice.nuie.nagoya-u.ac.jp/jikken/latex.html>

<sup>\*2</sup> <http://oku.edu.mie-u.ac.jp/~okumura/texwiki/>

<sup>\*3</sup> <http://www.math.sci.hokudai.ac.jp/~abenori/soft/abtexinst.html>

- Overleaf: <https://www.overleaf.com/>
  - Cloud LaTeX: <https://cloudlatex.io/>
- (4) レポートは覚え書きではない。他人に提出する報告書である。従って、実験の課題名、目的、使用器具、方法・手順、結果、考察（所見）、文献・資料をよく整理し、報告書（実験レポート）を一読してすべての内容が分かるように記述する。
- (5) レポートはそれ自体で完結した文書である。よって他の文献（特に実験指導書）を参考しなくとも内容が理解できるように記述する。ただし、道具の使用方法は記述しない。
- (6) 各課題ごとに、実験方法、実験結果、考察を記述する。
- (7) 実験方法・原理・アルゴリズム
- 分かりやすく記述する。実験で用いた装置の設計仕様や実験回路、原理図などは精確に描き、報告書に付ける。
- (8) 実験結果
- 実験に必要なプログラムは、プログラムリストを添付しコメントを付ける。そのプログラム（システム）の設計概念（考え方）を分かりやすく記述する。
  - 実験データは必要に応じて表・グラフにする。測定の際に読み取った生データは、有効数字が分かるように正しく書く。生データをもとに何らかの数値処理（情報処理）をしてグラフ化した場合には、その処理方法や理論も説明する。
  - グラフには、グラフの目盛、単位、目盛のスケール、パラメータを付け、実験データはグラフの各軸の目盛をもとにプロットする。グラフの曲線上のどこをとってもデータが正しく読み取れるようにする（有効数字が3桁なら3桁まで正確に読めるようにする）。
  - 表の上に、表番号と適切なキャプションを付ける。
  - 図の下に、図番号と適切なキャプションを付ける。
  - ハードウェア実験の場合は使用機器名、機器のシリアル番号、回路図を付ける。
- (9) 考察
- 何に対する考察なのか分かるように書く。
  - データの羅列や実験手順の説明を考察と勘違いしない。
  - 考察は他者と共有してはならない（他人のものを写してはならない）。
- (10) 参考文献
- レポートを書くうえで参考にした文献を記載せよ。
  - ウェブ・ページを記載する場合は適切な URL を示せ。Wikipedia のトップページを挙げるなどは論外である。
  - 実験指導書を参考文献に挙げてはならない。

### 3 レポート提出について

- (1) ウェブブラウザで TACT にアクセスし，ログインした後にコンピュータ科学実験のページを開く．  
`https://tact.ac.thers.ac.jp/portal/`
- (2) 左側メニューから課題を選択し，右側の課題一覧からレポート提出を行う課題を選択する．
- (3) レポートの PDF ファイルを提出する．ファイルサイズが 50MB を超えるとアップロードできないため注意すること．
- (4) レポートの提出日は，実験の終了日から一週間後とする．レポートの提出遅れは減点とする．
- (5) 再実験やレポートの再提出を求められた者は実験担当者の指示に従うこと．

### 4 その他

- (1) 本指導書の誤字・脱字・説明の誤りなどに気づいた場合，誤りの場所と内容を以下のメールアドレスに連絡していただきたい．  
`jikken-text@nuie.nagoya-u.ac.jp`

# ネットワークとセキュリティ

## 目次

<b>1</b>	<b>はじめに</b>	<b>1</b>
<b>2</b>	<b>ネットワーク概説</b>	<b>2</b>
2.1	概説	2
2.2	暗号化方式	7
2.3	各種サービス・プロトコル	8
2.4	ルータと静的ルーティング制御	18
<b>3</b>	<b>情報セキュリティ対策</b>	<b>21</b>
3.1	端末型接続と LAN 型接続	21
3.2	IP マスカレード	22
3.3	netfilter	22
3.4	iptables	24
3.5	フィルタリング・ルールの設定	26
3.6	アドレス変換の設定	31
3.7	ログ管理	32
<b>4</b>	<b>Linux システム概説</b>	<b>34</b>
4.1	Linux の誕生	34
4.2	デバイスとファイルシステム	36
4.3	Linux の起動プロセス	38
4.4	ネットワーク管理	40
4.5	CentOS のセキュリティ対策	43
4.6	パッケージ管理	50
4.7	ユーザ管理	50
4.8	プロセス管理	50
4.9	附録	51
<b>5</b>	<b>WWW サーバによる WEB アプリケーションプログラミング</b>	<b>53</b>
5.1	HTML と CSS	54
5.2	CGI と WSGI	57
5.3	SQL	59
<b>6</b>	<b>課題内容とレポート作成について</b>	<b>65</b>
6.1	実験スケジュール	65
6.2	実験課題	65
6.3	レポートについて	81
<b>A</b>	<b>サンプルリスト</b>	<b>84</b>
A.1	iptables 初期設定ファイル (iptables-sample.sh)	84
A.2	WSGI サンプル (sample.wsgi)	87

<b>B 不正アクセスの手口・被害</b>	<b>92</b>
B.1 不正アクセスの被害	92
B.2 不正アクセスの基礎知識	96
B.3 不正アクセス対策の基本技術	97
B.4 不正アクセスされた場合の対処方法	102
B.5 コンピュータウィルスが侵入したときの対処方法	102

作成者：松本哲也, 橋本健二, 駒水孝裕, 大平茂輝

改訂：2024 年 10 月 15 日 (松本)

改訂：2023 年 4 月 1 日 (橋本)

改訂：2022 年 4 月 1 日 (松本)

改訂：2021 年 4 月 1 日 (松本)

改訂：2020 年 4 月 1 日 (松本)

初版：2019 年 4 月 1 日

# 1 はじめに

情報通信ネットワークは、現代の様々な計算機利用において必要不可欠のものとなっている。

本実験では、TCP/IP ネットワークにおける各種サービスの提供・利用のために必要な基本的知識を学習することを目的とする。具体的には、各班ごとにローカルネットワーク・DMZ ネットワークを備えるサブネットネットワークを構築し、Linux におけるネットワーク構成方法、ファイアウォールの設置方法について学習する。次に、Python WSGI を用いて、SQL データベースと連携した WEB アプリケーションを作成し、ネットワークプログラミングの基礎を学習する。また、近年急激に被害が増大している計算機への不正アクセスについて、その実情や手口の事例を知るとともに、対策の基本技術について概説する。

本テキストの内容は、Python 言語、Linux システムの基本操作、資料の利用・作成方法は先行するプログラミング・演習科目で学習済みであることを前提とする。前提となる知識の修得について、本テキスト及び実験時間中に取り扱うことはしないので、予め復習しておくこと。



## 2 ネットワーク概説

### 2.1 概説

#### 2.1.1 インターネットについて

インターネットとは、大学や企業といった組織内で構築されたネットワークを、インターネットプロトコルという規格で相互接続した広域ネットワークである。プロトコルというのは通信を行うときのルールであり、インターネットプロトコルのことを IP (Internet Protocol) と略して呼ぶ。

2017 年 7 月の時点で、10 億 7497 万台の計算機がインターネットに接続されていると推計されている<sup>1</sup>。また、2018 年 1 月の時点で世界のインターネット利用人口が 40 億人 (53%相当) であると報告されている<sup>2</sup>。

日本では 1987 年に IP を用いたインターネットが構築された。総務省が 2018 年 6 月に発表した通信利用動向調査結果<sup>3</sup>によると、2017 年の人口普及率は 80.9 %、13～59 歳までの利用率は 9 割を超えている。端末別では、スマートフォンが 59.7%と最も多く、次いで自宅の PC が 52.5%であり、今回初めてスマートフォンが PC を上回った。スマートフォンについては 13～49 歳の 7 割以上が利用しており、50～59 歳の利用も 68.4%と PC の利用を上回っている。また、13～49 歳におけるソーシャルメディアの利用は 6 割を超えている。

#### 2.1.2 OSI 参照モデルとは

OSI 参照モデル (Open Systems Interconnection reference model) は、ISO (International Organization for Standardization: 国際標準化機構) や CCITT (Comite Consultatif International Telegraphique et Telephonique: 国際電信電話諮問委員会) などを中心となり制定された国際標準であり、異機種間のデータ通信を実現するためのネットワーク構造の設計方針である。OSI に基づいて、コンピュータの持つべき通信機能を階層構造に分割した 7 階層モデルを表 1 に示す。7 つの層は物理媒体に近い方から、物理層、データリンク層、ネットワーク層、トランスポート層、セッション層、プレゼンテーション層、そしてアプリケーション層と名付けられている。

#### 2.1.3 TCP/IP の起源

TCP/IP は、DOD (Department of Defence: アメリカ国防総省) の ARPA (Advanced Research Projects Agency: 高等研究計画局) のもとで構築された ARPANET (Advanced Research Projects Agency Network)<sup>4</sup>の通信プロトコルとして開発された。ARPANET は、当初パケット交換を使用して WAN (Wide Area Network: 広域網) を構築する実験のために 1970 年の初めに設計された。この実験は成功し、関連するプロトコル・スタック (プロトコルの組み合わせ) は年々拡張され、再定義されていった。また、この TCP/IP プロトコルは後に LAN (Local Area Network) にも適用されるようになった。

1980 年代の初めに、TCP/IP はバークレー版の UNIX 「4.2BSD」 (Berkeley Software Distribution) の重要な部分として実装され、TCP/IP はワークステーションの通信の実質的な標準として使用されるようになった。この結果 1983 年に、ARPANET に接続されたすべてのコンピュータとネットワークに TCP/IP プロトコルを使うことになった。

#### 2.1.4 TCP/IP プロトコルの概要

一般的に呼ばれている TCP/IP は、OSI の参照モデルの位置付けも含めて表 1 のような複数のプロトコルから構成されている。OSI の参照モデルが 7 階層のプロトコル層から構成されているのに対して、TCP/IP はアプリケーション層、トランスポート層 (TCP)、インターネット層 (IP)、ネットワーク・インタフェース層の概念的な 4 つの層から構成されている。

<sup>1</sup>ISC Domain Survey: <https://www.isc.org/network/survey/>

<sup>2</sup>DIGITAL IN 2018: <http://wearesocial.com/blog/2018/01/global-digital-report-2018>

<sup>3</sup>総務省 通信利用動向調査: [http://www.soumu.go.jp/johotsusintokei/statistics/data/180525\\_1.pdf](http://www.soumu.go.jp/johotsusintokei/statistics/data/180525_1.pdf)

<sup>4</sup>1973 年に DARPA (Defence Advanced Research Projects Agency Network) と改めた。

表 1: OSI 参照モデルと TCP/IP プロトコルモデル

OSI 参照モデル (7 階層モデル)			TCP/IP モデル (4 階層)
第 7 層	アプリケーション層	どのような通信サービスを行い、何を実現するか？ (アプリケーションの種類に関する規定)	アプリ ケーション層
第 6 層	プレゼンテーション層	どのような表現形式で送るのか？ (データの種類や送信ビット数に関する規定)	
第 5 層	セッション層	どのような対話モードで送るのか？ (通信モードや同期方式に関する規定)	
第 4 層	トランスポート層	相手に正確に届いたかどうかの確認方法は？ (送受信確認やアプリケーションの識別に関する規定)	トランスポート層 (TCP/UDP 層)
第 3 層	ネットワーク層	相手の識別アドレスは？通信網をどう使うのか？ (通信経路の選択や識別アドレスに関する規定)	インターネット層 (IP 層)
第 2 層	データリンク層	伝送路の確保と端末の識別方法は？ (通信路の確保やエラー訂正に関する規定)	ネットワーク インタフェース層
第 1 層	物理層	伝送路に情報を送る媒体、方法は？ (物理的な回線や機器類、電気信号に関する規定)	

### 2.1.5 TCP/IP とイーサネット

イーサネット (Ethernet<sup>5</sup>) は、Xerox PARC (Palo Alto Research Center) で 1973 年に考案されたコンピュータネットワークの規格であり、その後の開発と標準化作業を経て 1983 年に策定された IEEE 802.3 が、現在のイーサネットの標準となっている。IEEE 802 では、データリンク層を上位の LLC 副層と下位の MAC 副層とに分けており、イーサネットの基本仕様は、OSI 参照モデルの第 1 層 (物理層) と第 2 層 (データリンク層) の下位の MAC 副層で規定されている。TCP/IP モデルでは最下層のネットワーク・インタフェース層に相当する。

イーサネットでは、データの伝送単位をパケットではなくフレームと呼ぶ。フレームには、データの宛先アドレスと送信元アドレスを記述するフィールドがあり、アドレスとしてインタフェースの MAC アドレスを設定する。MAC アドレス (Media Access Control address) は、ネットワーク上で各ノードを識別するために、ネットワークカード (NIC) などのネットワーク機器ごとに割り当てられるハードウェア固有のアドレスである。イーサネットの場合、48 ビットの符号で表現され、上位 24 ビットは IEEE が管理・割り当てを行っているネットワーク機器のベンダコードであり、OUI (Organizationally Unique Identifier) と呼ばれる。下位 24 ビットは、各ネットワーク機器ベンダが自社製品に対して独自に重複しないよう割り当てている。この仕組みにより割り当てられる MAC アドレスは、原則として世界で唯一の番号となる<sup>6</sup>。

通常、MAC アドレスは、xx:xx:xx:xx:xx:xx や xx-xx-xx-xx-xx-xx のように各オクテット<sup>7</sup>をコロンやハイフンで区切った 16 進数で表現するのが一般的である。MAC アドレスを確認するには、Windows の場合 ipconfig コマンドを利用し、Linux の場合は ip コマンドを利用する。

<sup>5</sup>“ether” は 19 世紀に光や力を伝える媒体として空間に充満していると考えられた仮想的な物質「エーテル」にちなんでいる

<sup>6</sup>実際はどうか興味がある人は調べてみるとよい

<sup>7</sup>8 ビット単位の情報

```
C:\> ipconfig /all

Windows IP Configuration

(略)

Ethernet adapter ワイヤレス ネットワーク接続:

    Media State . . . . . : Media disconnected
    Description . . . . . : Intel(R) PRO/Wireless 2915ABG Network Connection
    Physical Address. . . . . : 00-12-F0-5C-7D-A5

Ethernet adapter ローカル エリア接続:

    Connection-specific DNS Suffix . : nagao.nuie.nagoya-u.ac.jp
    Description . . . . . : Broadcom NetXtreme Gigabit Ethernet
    Physical Address. . . . . : 00-11-25-15-52-55
```

```
# /usr/sbin/ip addr
eth0: (略)
    link/ether 00:11:25:15:52:55

(略)
```

MAC アドレスの上位 3 オクテットで表される OUI を登録した組織は、IEEE のサイト<sup>8</sup>で検索することができる。上述の例の場合、上位 3 オクテットは“0012F0”と“001125”であるため、これを入力して検索すると次のような結果が得られる。

Assignment	Assignment Type	Company Name	Company Address
00-12-F0 (hex) 0012F0	MA-L	Intel Corporate	Lot 8, Jalan Hi-tech 2/3 Kulim Kedah 09000 MY

Assignment	Assignment Type	Company Name	Company Address
00-11-25 (hex) 001125	MA-L	IBM Corp	3039 E Cornwallis Road Research Triangle Park NC 27709-2195 US

MAC アドレスと IP アドレスの相互変換には、ARP や RARP というプロトコルを用いる (2.3.13 章参照)。

## 2.1.6 TCP/IP プロトコルの具体的な内容

### (1) IP (Internet Protocol)

インターネット・プロトコルは OSI のネットワーク層に相当する。ネットワーク層を利用してデータグラムを転送するのが主な役割である。IP には次のような機能がある。

- a) パケットのフラグメンテーション (分割)/再組立
- b) アドレッシング (ホスト, 端末の番地割り当て)
- c) ルーティング (通信経路の選択)
- d) エラー制御

<sup>8</sup>IEEE Standards Association Registration Authority: <https://regauth.standards.ieee.org/standards-ra-web/pub/view.html>

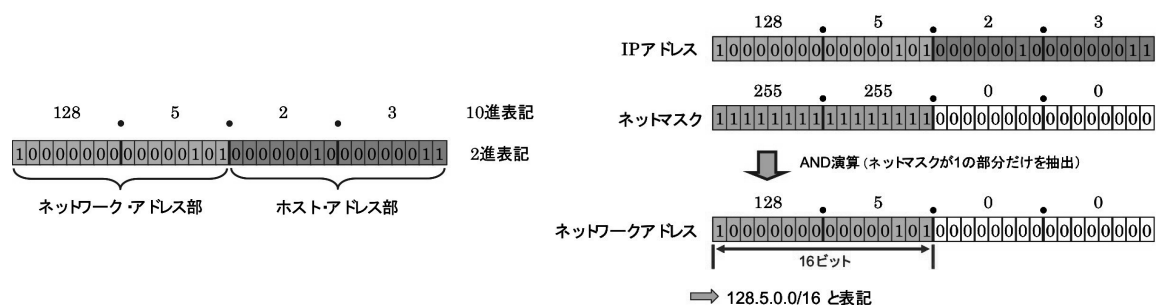


図 1: IP アドレス表記とネットマスク

#### a) フラグメンテーション/再組立

TCP/IP はパケット (1 バイト又は何バイトかのデータの固まり) に基づいた通信を行う。パケットは、データについての情報部分であるヘッダ (先頭) 部分とデータ部分から構成されている。ヘッダとデータは各プロトコル層でそれぞれ異なる。

アプリケーションがデータを送信する際、TCP 処理モジュールが TCP ヘッダを付加して IP 処理モジュールに渡す。IP 処理モジュールは、さらに IP ヘッダを付加する。このデータがネットワーク・インタフェース (データリンク層・物理層) のモジュールに渡され、例えば Ethernet ヘッダが付加される。ここでフレーム検査シーケンス (FCS:Frame Check Sequence) の計算を行い、FCS を付加する。ただし、アプリケーションのデータが大きい場合、複数の IP パケットに分割 (フラグメンテーション) される。これは、再送信などによるネットワークの負荷軽減を考慮しているからである。Ethernet においては、最大パケット長は 1500 バイトであるから、これよりも大きいデータは、1500 バイト以下のパケットに分割されるわけである。

#### b) アドレッシング (番地割当)

どのような通信システムでも、そこに接続されたパソコンやワークステーションなどはすべて通信が可能であることが要求される。このため、インターネット上の装置やホストには個々のアドレスが割り当てられる。これをインターネット・アドレス (IP アドレスともいう) というが、インターネット・アドレスは 32 ビットのアドレス・フィールドを使用する。アドレス・フィールドのビットは、0 から 31 まで番号付けられている。このフィールドは、ホスト自身を識別する部分 (ホスト・アドレス部) とホストが存在するネットワークを識別する部分 (ネットワーク・アドレス部) の二つの部分に分けられる。ネットワーク・アドレス部とホスト・アドレス部の識別には、ネットマスクと呼ばれる 32 ビット幅の数値が使用される。ネットマスクとは、IP アドレスからネットワーク・アドレス部を抽出するためのマスク値である。IP アドレスからネットワーク・アドレス部を取り出したり、ホスト・アドレス部だけを取り出したりするには、このネットマスクと IP アドレスとの AND 演算を行えばよい。また、インターネット・アドレスには 4 つのクラスがある。それぞれのクラスはユニークなビットパターンから始まっているため、容易に識別できるようになっている。インターネット・アドレスは、128.5.2.3 のようにドットで区切られた 4 つの 10 進数で定義される。32 ビット・アドレスを 4 つのオクテット (1 オクテットは 8 ビット) と呼ばれる 8 ビット・フィールドに分割し、それを 10 進数で表現する。図 1 に IP アドレスの例を示す。また、ホスト部のビットがすべて 1、または 0 になっているものは、特殊用途 (ブロードキャストアドレス) に使用される。

有効なネットワーク番号は各クラスで表 2 のようになる。‘hhh’ はホスト・アドレスになる。また、IP アドレス A, B, C, D 各クラスのビット構成を表 2 に示す。

IP アドレスの各クラスごとに、デフォルトのネットマスクが決まっており、クラス A は 24 ビット、クラス B は 16 ビット、クラス C は 8 ビットとなっている。これをさらに拡張するものとして、ホスト・アドレス部を 2 つに分割することで、より柔軟にネットワークを構築できるサブネット分割という方法が用意されている。サブネット分割では、ホスト・アドレス部をサブネット・アドレス部とサ

表 2: 4 つのクラスと IP アドレスの範囲

クラス A	001.hhh.hhh.hhh	～	126.hhh.hhh.hhh
クラス B	128.001.hhh.hhh	～	191.254.hhh.hhh
クラス C	192.000.001.hhh	～	223.255.254.hhh
クラス D	224.000.000.000	～	239.255.255.255

表 3: プライベートアドレスの範囲

クラス A	10.0.0.0	～	10.255.255.255
クラス B	172.16.0.0	～	172.31.255.255
クラス C	192.168.0.0	～	192.168.255.255

ブネット内のホスト・アドレス部とに分割する。サブネット分割した場合のネットマスク値は、ネットワーク・アドレス部とサブネット・アドレス部の両方を含み、これをサブネットマスクと呼ぶ。

すべてのインターネット・アドレスは、SRI (Stanford Research Institute) インターナショナルの NIC (Network Information Center) で管理されている。インターネットに接続されるコンピュータは、それぞれ異なるインターネットアドレスを持ち、これをグローバルアドレスという。しかし、すべてのコンピュータにグローバルアドレスを割り振ることは困難である。そこで、直接インターネットに接続しないコンピュータのために、表 3 のような IP アドレスが用意されている。この IP アドレスを、グローバルアドレスと区別してプライベートアドレスと呼ぶ。

#### c) ルーティング (通信経路の選択)

ルーティングとは、任意の宛先のホストにデータを転送する際に、複数の到達可能な通信経路 (ルート) のなかから一つの通信経路を選択することである。通信経路を選択するアルゴリズムには距離の違いによるものと、リンクの状態 (接続状態) によるものの二つがある。前者の代表が RIP (Routing Information Protocol : 経路情報プロトコル) であり、後者の代表が OSPF (Open Shortest Path First) のルーティング・プロトコルである。

ルータは、パケットをどの通信経路で転送するかを判断するために、ルーティング・テーブルと呼ばれるテーブルを使用する。もし宛先のホストがネットワークに直接に接続されているならば、ルータは他のルータを使用せずにパケットを転送することができるが、そのホストが遠隔地のネットワーク上にある場合は、最終的に接続されるホストの宛先に近い他のルータにパケットを送信しなければならない。遠隔地のネットワークの通信経路は、固定的 (スタティック) に設定したり、RIP や OSPF のようなプロトコルを使用して、回線状態によって接続状態を変えるダイナミック・ルーティングを設定できるようになっている。

#### d) エラー制御

IP (Internet Protocol) は、信頼性のある通信機能を提供していない。つまり確認応答、データの紛失またはデータ障害による再送、フロー制御、そしてパケットの送受信の際に、パケットの到着順序をそろえるなどの処理をする必要がある。

### (2) トランスポート層プロトコル (TCP / UDP)

トランスポート層のプロトコルは、OSI 参照モデルのトランスポート層と同じで、あるアプリケーションから他のアプリケーションへの通信を提供する。また、トランスポート層には二つの異なるサービスを行うプロトコル TCP (Transmission Control Protocol), UDP (User Datagram Protocol) が用意されている。IP がインターネット・アドレスをもとにパケットの交換を行うのに対して、TCP, UDP はポートという概念でパケットの交換



表 4: TCP と UDP の違い

	UDP	TCP
接続形態	1:1 および 1:n どちらも可能	1:1 のみ
アプリケーションの特定方法	UDP ポート番号	TCP ポート番号
送受信の単位	パケット	ストリーム
宛先までの到達保証	なし	あり
送信エラー時の動作	パケット破棄	自動的に再送
事前のアプリケーション同士の接続動作 (コネクションの確立)	不要	必要
処理の重さ	軽い	重い

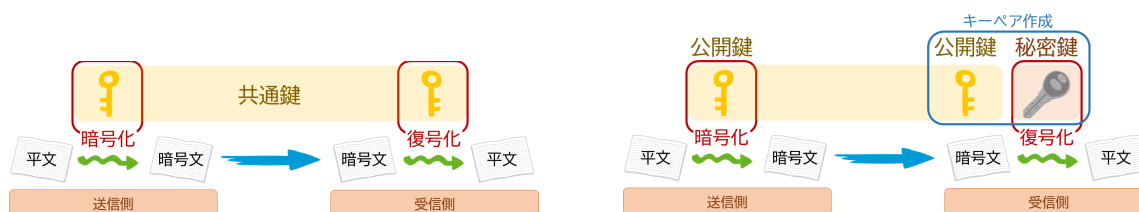


図 2: 共通鍵暗号方式 (左) と公開鍵暗号方式 (右)

を行う。IP で確立された通信路の上にポートという出入口を持った通信路が仮想的に複数確立される。TCP と UDP の違いをまとめたものを、表 4 に示す。

UDP の本来の目的は、IP プロトコルが持つ最低限の機能をユーザアプリケーションに提供することにある。そのとき、TCP の場合と同様にポート番号を用いることで、複数の相手と同時に通信が可能になる。さらに、TCP のような信頼性を提供していない分、UDP プロトコルモジュールの処理が単純になり、TCP に比べて高速なデータ転送が可能になる。また、TCP のようなコネクションの確立を行わないので、あらかじめブロードキャストアドレスを決めておけば、一度に同一ネットワークのすべてのコンピュータに、パケットを送ることができる。

## 2.2 暗号化方式

ネットワーク上を流れるデータの盗聴は、ネットワークに関する多少の知識とツールを用いることで比較的容易に行えてしまうことから、個人情報やパスワードなどの重要な情報を暗号化する技術は不可欠である。

### 2.2.1 共通鍵暗号方式

暗号と復号に同じ鍵（共通鍵）を用いる方式を共通鍵暗号方式と呼ぶ（図 2 左）。

共通鍵暗号方式では、暗号化されたデータを復号化する前に相手に鍵を渡す必要がある。二者間でのみ鍵を共有する方式であり、通信相手ごとに共通鍵を作成する必要があるため、共通鍵の煩雑な管理と安全な配布が課題であるが、処理速度は早いのが特徴である。鍵の配布時に第三者に盗まれてしまう危険性を鍵配送問題と言う。

代表的な共通鍵暗号方式として、DES (Data Encryption Standard)、トリプル DES、AES (Advanced Encryption Standard) が挙げられる。現在は、DES に代わる方式である AES が主流となっており、無線 LAN 上で通信を暗号化して保護する規格である WPA2 (Wi-Fi Protected Access 2) の暗号化方式 (WPA2-AES) としても採用されている。

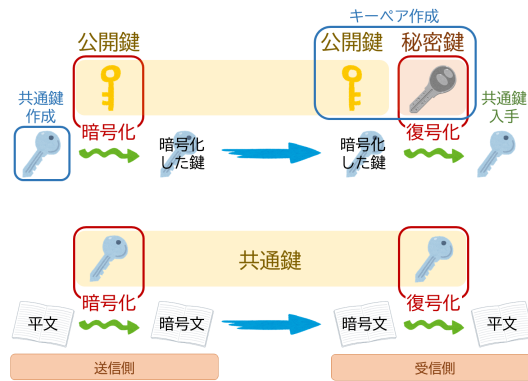


図 3: ハイブリッド暗号方式

## 2.2.2 公開鍵暗号方式

暗号用の鍵（公開鍵）と復号用の鍵（秘密鍵）という2つの異なる鍵を用いる方式を公開鍵暗号方式と呼ぶ。公開鍵と秘密鍵はペアで利用し、公開鍵はネットワーク上で広く公開するが、秘密鍵はキーペアの所有者が厳重に保管する。

公開鍵暗号方式では、送信側は受信側が公開している公開鍵を使ってデータを暗号化して送信し、受信側は自身の持つ秘密鍵を用いてデータの復号化を行う（図2右）。公開鍵は誰でも取得可能だが、暗号化したデータを復号できるのは公開鍵とキーペアになる秘密鍵を持つ者だけであるため、通信相手ごとに鍵を作成する必要はなく鍵の管理は容易である。しかし、共通鍵に比べると処理速度が遅いのがデメリットである。

公開鍵暗号方式は、鍵生成・暗号化・復号の3つのアルゴリズムで構成され、代表的な公開鍵暗号方式として、RSA暗号化方式がある。RSA暗号化方式には、前述した公開鍵による暗号化と秘密鍵による復号化という基本的な性質とは別に、秘密鍵によって変換したデータを公開鍵によって元に戻すことができるという特徴がある。RSA暗号化方式のこの特徴を利用したものに電子署名がある。

## 2.2.3 ハイブリッド暗号方式

共通鍵暗号方式で用いられる共通鍵は渡す際に傍受される恐れがあるため、共通鍵の受け渡し手段として公開鍵暗号を用いることにより、通信の安全性を保つのがハイブリッド暗号方式である（図3）。

ハイブリッド暗号方式では、公開鍵暗号方式と同様に受信側で公開鍵と秘密鍵のキーペアを作成する。送信側は作成した共通鍵を公開鍵で暗号化してから受信側に渡し、受信側は暗号化された共通鍵を秘密鍵で復号化して共通鍵を入手する。以降は、共通鍵暗号方式と同様に、共通鍵を用いてデータのやりとりを行う。

SSL/TLS通信では、ハイブリッド暗号方式が電子署名/電子証明書と組み合わせて利用される。SSL/TLSについては、4.5.2項で詳しく述べる。

## 2.3 各種サービス・プロトコル

### 2.3.1 クライアントサーバモデル

ネットワークを利用してサービスを提供するシステムにおいては、エージェントモデルやP2Pモデルなど幾つかの方式が用いられているが、その多くはクライアントサーバモデルで構成されていると考えてよい。クライアントサーバモデルは、何かのサービスを提供するプログラム（サーバ）と、サーバに処理を依頼し結果を受け取るプログラム（クライアント）で構成されている（図4）。

クライアントサーバモデルの利点としては、互いの満たすべき約束事（仕様）をあらかじめ決めておくことにより、個々の開発を独立して行うことができ、開発の負担を分散できるということがある。また、クライアントサーバモデルによってネットワークを構成することにより、サービスの提供に必要な資源の分散や共有を柔軟に行うことができる（図5）。

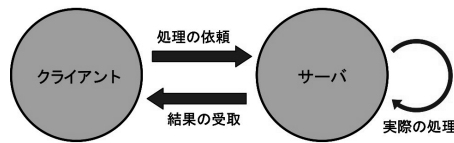


図 4: クライアントサーバモデル

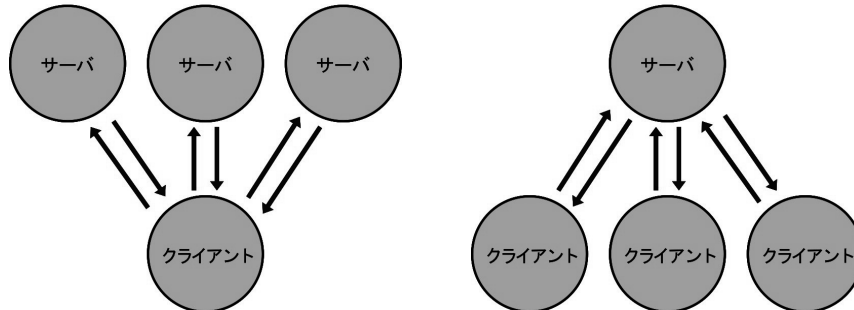


図 5: サービスの分散 (左) と共有 (右)

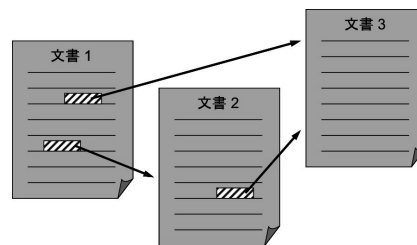


図 6: ハイパーテキスト

### 2.3.2 WWW サービス

WWW (World Wide Web) は、インターネット上のリンクを持つ文書へアクセスするためのアーキテクチャの枠組みである。文書中のある箇所から別の箇所への接続 (リンク) を可能とする文書の仕組みをハイパーテキスト (hypertext) と呼ぶ (図 6)。

WWW は典型的なクライアントサーバモデルのサービスである。クライアントの資源の要求 (リクエスト) に対し、サーバはその要求を解釈して対応する資源を返す。また要求に応じられない場合にはエラーメッセージを返す。これを応答 (レスポンス) と呼ぶ。

サーバとのやり取りには、様々なプロトコルが用いられる。代表的なプロトコルには、HTTP/HTTPS, FTP などがある。WWW では、URL と呼ばれる、プロトコル等の資源取得手段を示すスキーマを含めた識別子を用いて資源を扱う。たとえば次のような URL を用いる。

```
http://www.ice.nuie.nagoya-u.ac.jp/jikken/index.shtml
```

URL はスキーマと資源を特定する文字列をコロン「:」で結んだものと定義されている。上の例は http というスキーマを持つ //www.ice.... という文字列で特定される資源を示す URL である。多くの場合、スキーマはサーバとのプロトコルと対応する。必要な資源を特定する方法はプロトコルによって異なるものであり、必然的にコロン以降の文字列の定義はスキーマに固有のものとなる。

一方、インターネットに接続された計算機は、階層構造をもったファイルシステム上に文書やプログラム等の資源をファイルとして保持することを想定して、コロン以降の文字列に関しては、計算機名とファイルのパスで資源が特定できるという定義を http では利用している。具体的には連続する斜線 (スラッシュ) // に続けて資源を保持するサーバ名とサーバ上の資源の階層構造上の位置を示すパスを斜線で区切って次のように用いる。



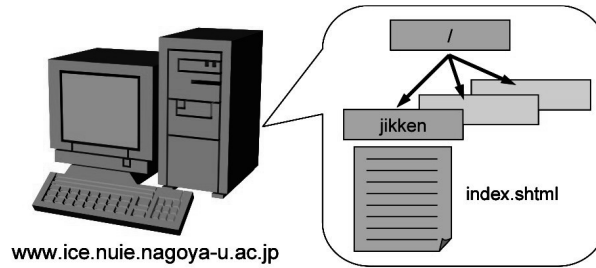


図 7: `http://www.ice.nuie.nagoya-u.ac.jp/jikken/index.shtml`

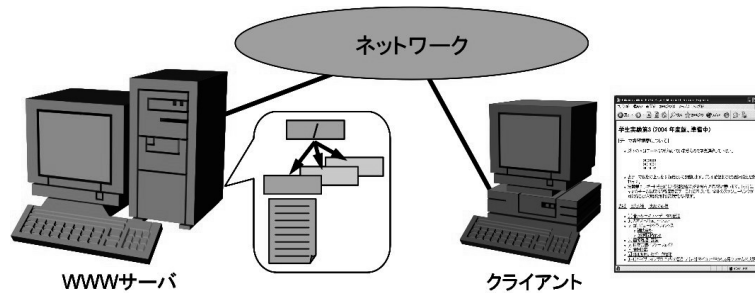


図 8: WWW サービス

//サーバ名/パス

UNIX のディレクトリ構造に倣い、パスの階層構造は / を区切り文字に用いて表す。つまり、先に示された URL の例は `www.ice.nuie.nagoya-u.ac.jp` という名前を持ったサーバが保持する `jikken` ディレクトリ下のファイル `index.shtml` を示すものということになる (図 7)<sup>9</sup>。

WWW クライアントは、WWW のハイパーテキストを実現するため、HTTP/HTTPS や FTP 等を介して資源を取得し、利用者へ提示する。

WWW が対応しているプロトコル群の中でも HTTP/HTTPS は中心的なプロトコルである。WWW は HTTP/HTTPS を中心に発展してきたため、WWW サーバとは、HTTP/HTTPS を介して資源を提供するプログラムあるいはコンピュータを指す。

コンピュータ科学科計算機システム (旧情報工学コース教育用計算機室) の端末では、WWW サービスのクライアントプログラムとして、mozilla 等の複数の WWW ブラウザと呼ばれるプログラムが利用できる。WWW ブラウザは WWW サービスを通じて得られる HTML 文書の解釈を行いユーザの閲覧を助けるためのインタフェースの提供を主たる目的とするクライアントプログラムである。

### 2.3.3 FTP サービス

WWW サービスで利用されるスキーマのうち、`http/https` に次いで頻繁に用いられるのは `ftp` だろう。スキーマの名前通り、`ftp` で利用される通信手順 (プロトコル) は FTP (File Transfer Protocol) と呼ばれる。ソフトウェアの配布やデータアーカイブの配布など、ファイル転送のサービスもインターネットの重要なサービスの一つである。このファイル授受のためのサービスを担うのが FTP である。

FTP サービスの主な利用法は、サーバの保持するファイルをクライアントへ、またはその逆にクライアントの保持するファイルをサーバへコピーするというものである。ファイル操作の一環として、ファイルの削除やファイル名の変更、ディレクトリの管理等の機能も備えている。通常の FTP では、アカウントが登録されている利用者しかファイル転送できない。そこでオープンソースなどのファイルを不特定多数に配布するため、アカウントの無いユーザに対して特定のファイルを公開する仕組みとして、Anonymous FTP が用意されている。上述の `ftp` スキーマを持つ URL への WWW ブラウザを用いたアクセスでも匿名アカウントが暗黙のうちに利用されている。

<sup>9</sup>詳しくは RFC1728 を参照すること。

FTP サービスのクライアントプログラムとして、コンピュータ科学科計算機システムの端末では、ftp が利用できる。

#### 2.3.4 電子メールサービス

電子メールは、最も古いネットワークアプリケーションの一つである。インターネット上で用いられているプロトコルは、メール配送のための SMTP (Simple Mail Transfer Protocol) と、メールボックスから到着したメッセージを取り出す POP3 や IMAP4 などである。メールボックスサービスのサーバはユーザ宛に配送されたメール文書を保持し、要求にしたがってクライアントプログラムへ送信する。メール文書はユーザに固有のものであり、FTP サービスの場合と同様に、その受信に際してユーザ名と認証のためのパスワードを必要とする。

メールの配送は、UA (User Agent) と MTA (Message Transfer Agent) で構成されている。UA は、私たちが普段利用しているメールソフトウェア (メーラ) であり、利用者とのインタフェースを司っている。MTA は電子メールの配送を司っており、複数の MTA が協調して配送を行っている。メールの配送手順を次に示す。

- (1) 発信者の UA から近くの MTA に SMTP によってメッセージと宛先が渡される
- (2) MTA (Sender SMTP) は、受取先となる MTA (Receiver SMTP) と通信を行いメッセージを送る。  
直接受取先 MTA と通信できない場合は中継 MTA に送る。
- (3) MTA (Receiver SMTP) は、受取人のメッセージボックスに書き込む。

コンピュータ科学科計算機システムの端末では、電子メールサービスのクライアントプログラムとして、mail コマンドや evolution が利用できる。

#### 2.3.5 DNS [17][18]

DNS (Domain Name System) は、ホスト名と IP アドレスの対応情報を提供する。ホスト名と IP アドレスとの対応情報はゾーンと呼ばれる単位で分散管理され、ゾーンは階層型に構成される。

DNS はインターネット上にある全世界のホストのホスト名と IP アドレスを管理できる。また、LAN 内の閉じられたシステムとして構築することもできる。

ゾーンの情報を管理、提供するのが DNS サーバである (ネームサーバとも呼ばれる)。DNS サーバはホスト名を IP アドレスに変換するサービスを提供する。これは正引きと呼ばれ、ネットワークアプリケーション (メールツール、Web ブラウザ、FTP 等) でホスト名を指定した時に利用される。また、IP アドレスをホスト名に変換するサービスを提供する。これは逆引きと呼ばれ、netstat, tcpdump 等から利用される。

この他に MX レコードと呼ばれる、ドメイン名をメールサーバ名に対応付けるサービスを提供する。MX レコードはメール配送プログラム (MTA) から利用される。

DNS サーバが提供するサービスを受けるのが、DNS クライアントである。DNS クライアントから DNS サーバへのアクセスは、ネットワークアプリケーションに組み込まれているリゾルバと呼ばれるライブラリルーチンが、設定ファイル/etc/resolv.conf に記述された DNS サーバの IP アドレスを得て行う。

DNS の問い合わせと応答でやり取りされるデータはたいてい 1 パケットに収まることから、リゾルバと DNS サーバ間では主に UDP による通信が行われ、応答データが 512 オクテットを超えたり DNS サーバ間でゾーン転送<sup>10</sup>を行う場合には、信頼性の高い TCP による通信が行われる。使用するポート番号は 53 である。

#### 2.3.6 DHCP サービス [17][18]

ネットワークにコンピュータを接続するには、自身の IP アドレスやサブネットマスク、デフォルトゲートウェイのアドレス、DNS に関する情報を正しく設定する必要がある。しかし、これらの情報を管理者から聞いて手作業で設定するのは面倒であり、ミスも生じやすい。

DHCP (Dynamic Host Configuration Protocol) は、IP アドレスをホストに自動的に割り振るサービスである。また、ネットマスク値や DNS ドメイン名等のネットワーク情報のサービスも行う。

<sup>10</sup>プライマリ DNS サーバでゾーン情報が更新されたときに、セカンダリ DNS サーバがゾーン情報を取得する仕組みのことである。

DHCP サーバは前もって設定された範囲の IP アドレスをプールし、ホストから要求があった時、その中から未使用のアドレスを自動的にホストに割り振る。IP アドレスにはリース期間があり、ホストから要求がないままリース期限を過ぎたものは回収される。

サーバとして運用されるようなコンピュータに対しては、IP アドレスと接続元のコンピュータの MAC アドレスを対にして管理することにより、常に決まった IP アドレスを割り当てることが可能となる。

DHCP は UDP 上で動作するプロトコルである。使用するポート番号は、DHCP サーバ宛にパケットを送る場合が 67 であり、DHCP クライアント宛にパケットを送る場合が 68 である。

DHCP クライアントはブロードキャストによってサーバを探す。このため原則として 1 つのネットワークに DHCP サーバは 1 台だけ用意する。

### 2.3.7 Telnet プロトコル

前節までに説明された HTTP, FTP, SMTP, POP 等といったネットワークサービスで利用されるプロトコルは TCP/IP 通信の上で定められた行指向の手順である。クライアントプログラムを介したサービスの利用においては、ユーザの利便性のために実際のプロトコルの様子は隠蔽されている。そこで、ここではネットワークプログラムのための基礎知識の修得のために、プロトコルの実際を `telnet` コマンドを用いて確認する。

なお、本章では具体的なプロトコルの確認方法についてだけ述べることにする。各プロトコルの詳細について知りたい場合は、RFC 文書や参考書等を読むことを推奨する。`telnet` コマンドについては、`man` コマンドによってマニュアルページを参照すること<sup>11</sup>。基本的な `telnet` コマンドの利用方法は次の通りである。

```
% telnet 

|      |       |
|------|-------|
| ホスト名 | ポート番号 |
|------|-------|


```

これにより、指定されたホスト名に対応する計算機との通信を、指定されたポート番号を通じて開始することができる。ホスト名の代わりに IP アドレス、ポート番号の代わりにサービス名を指定することもできる。

例えば、サービス名に `http`, `ftp`, `smtp` や `pop3` を指定することで、それぞれ HTTP, FTP, SMTP, POP の標準的なポート番号を指定したのと同様の通信を開始することができる。これは、代表的なポート番号に関して、サービス名との対応データベースを参照して実現されている機能であり、UNIX 類似の OS では `/etc/services` に対応表にあたるファイルが用意されている。したがって、対応表に存在しないサービス名を用いることはできないし、ホスト名に指定した計算機のサーバプログラムが対応表とは異なるポート番号を利用していた場合には期待する結果が得られないので注意する必要がある。

`telnet` コマンドを用いて確認することのできるプロトコルは IP ネットワーク上で TCP を用いて構築されているものである。IP ネットワークでやりとりされる IP パケットの発信元・受信先アドレスは IP アドレスである。`telnet` コマンドにおいて受信先として指定されたホスト名から IP アドレスを特定する仕組みが必要となる。

ホスト名と IP アドレスの対応に用いられるシステムには様々なものが存在する。そのうち、インターネット上で一意に定まるホスト名であるところの、FQDN の特定には DNS というデータベースの利用が必要となる。UNIX 類似の OS では `nslookup` コマンドや `host`, `dig` コマンドを利用して、その OS で利用しているホスト名と IP アドレスの対応関係を調べることができる。

### 2.3.8 HTTP (HyperText Transfer Protocol) と HTTPS

HTTP サーバは、接続してきたクライアントからのリクエストを受信し、対応する応答を送信する。サービスは TCP 通信を用いて提供されるので、`telnet` コマンドを用いて動作の確認を行うことができる。HTTP サーバで標準的に利用されるポート番号は 80 番なので、以下のように実行すればよい。

```
% telnet 

|                     |
|---------------------|
| サーバのホスト名または IP アドレス |
|---------------------|

 80
```

基本的に、通信を確立しクライアントのリクエストに対して応答を送信すると、サーバ側から通信を終了させる。

HTTP には異なるバージョン番号を持つものはいくつかあり、新旧のプロトコル自体に非互換な部分が存在する。ただし、各バージョンの規定において若いバージョン番号を持つプロトコルに対応することが義務づけられ

<sup>11</sup> コマンドプロンプトから `man telnet` とすれば良い。

ており、実際に新しいプロトコルに対応するサーバプログラムのほとんどが古いプロトコルにも対応しているので、実用的には問題は無い。現状では、0.9, 1.0, 1.1, 2 の 4 通りの規定が維持されている。

それぞれの詳細は RFC1945 (HTTP/0.9, HTTP/1.0), RFC2616 および RFC7230～7235 (HTTP/1.1), RFC7540 (HTTP/2) を参照すること。

**HTTP/0.9** HTTP/0.9 では、通信確立後、サーバはクライアントからのリクエストを待ち受ける。クライアントは要求するリソースを示す URL もしくはパスを含む次のような 1 行のリクエストを送信するものとされる。

GET URL またはパス

例)

```
% telnet www.ice.nuie.nagoya-u.ac.jp 80
Trying 172.16.1.7...
Connected to www.ice.nuie.nagoya-u.ac.jp.
Escape character is '^]'.
GET /index.shtml
```

GET の後の空白に続けて指定される文字列は返信を期待するリソースを示す URL <sup>12</sup>そのもの、あるいは URL からスキーマを指定する部分を省いたパスを用いる。

telnet コマンドでは、リクエストに対応する文字列を入力した後に続けて改行をすることで上述の規定に沿ったリクエストが送信される。また、サーバが返信するデータは、リクエスト末尾の改行に続けて、その内容が表示される。サーバの応答の直後、通信はサーバ側から切断される。

**HTTP/1.0** HTTP/1.0 では、クライアントの送信するリクエストには複数のメソッドが規定され、HTTP/0.9 のリクエストに対応する機能は GET メソッドにより実現される。HTTP/1.0 のリクエストは次のような、メソッド、要求するリソース、HTTP のプロトコルバージョンを特定する文字列を空白で区切った 1 行の文字列で始まる。

メソッド URL またはパス HTTP バージョン

例)

```
HEAD /index.shtml HTTP/1.0
空行
```

標準のメソッドとして、GET, HEAD, POST メソッドの 3 種が規定され、メソッドを特定する文字列には、それぞれ GET, HEAD, POST が用いられる。HTTP のプロトコルバージョンを示す文字列には HTTP/1.0 が定められる。古いプロトコルのリクエストはプロトコルバージョンを示す文字を欠いた GET メソッドということになり、文字列 HTTP/1.0 の有無で区別されることとなる。

2 行目以降はヘッダ情報が記述され、空行によりヘッダ情報の終了がサーバに知らされる。クライアントは、メソッドおよびヘッダ情報の内容によっては、続けてデータの送信を行う。データの送信を行う場合は、ヘッダ情報によって、続けて送信されるデータの長さがサーバに知らされているので、サーバは指定された分量のデータを受信し、クライアントのリクエストに対する応答を開始する。

サーバの送信する応答も、リクエストの 2 行目以降と同様に、ヘッダ情報を示す部分とデータ本体とからなり、両者は空行で区切ることが規定されている。

HEAD メソッドはサーバからの応答がヘッダ情報に限られている点だけが GET メソッドと異なる。データ本体は不要でリソースの情報のみを取得したい場合に用いられる。例えば、リソースの存在や更新状況の確認等の用途が考えられる。

<sup>12</sup>厳密には RFC1738 の規定する URL ではなく、より一般化された URI と定められているが、ここでは URL と同一視する。



HTTP/1.0 では、GET、HEAD、POST の 3 種の標準メソッド以外に PUT、DELETE、LINK、UNLINK の拡張メソッドが規定されている。拡張メソッドについては RFC 文書を参照すること。

**HTTP/1.1** HTTP/1.1 では、リクエストのヘッダ情報として Host ヘッダの送信が必須となる。これは、1 台の計算機で複数のホスト名を持ち、別々の階層構造のリソースを保持するためのものである。さらに、1 リクエストごとに TCP 接続を確立しなくて済むように、明示的に閉じることを指定しない限り接続を維持する機能が採用されている。そのため、レスポンス終了後に接続を閉じる場合には、Connection ヘッダに close 値を与える必要がある。

例)

```
HEAD /index.shtml HTTP/1.1
Host: www.ice.nuie.nagoya-u.ac.jp
Connection: close
空行
```

また、HTTP/1.1 では HTTP/1.0 のメソッドに加えて OPTIONS、TRACE、PATCH の 3 種の新たなメソッドが規定されている。OPTIONS メソッドはリソースに対して利用可能なメソッドの種類を取得するためのものである。また、既存のメソッドにおけるリソースを指定する文字列以外に、\* (アスタリスク) 1 文字からなるサーバの保持する全てのリソースに一致する指定を用いることができる。利用可能なメソッドの一覧は、リクエストに対する返信中の Allow ヘッダにより得られる。OPTIONS 以外の新たに規定されたメソッドに関しては RFC 文書を参照すること。

HTTP/1.1 では、新規のメソッド以外にも、行指向でないプロトコルや通信の維持による効率的なデータの送受信方法の導入や、代理アクセスに関する規定等の他に、リクエストに対する応答やリクエストに付随してクライアントが送信するデータ本体に関するより明確な規定がなされている。様々な拡張は多くのサーバプログラムの実装で行われた多様な拡張を取り込むものである。

HTTP/1.1 の問題点として、1 つずつしかリクエストを処理できないことによる非効率性が上げられる。現在のように、大量の画像や CSS、スクリプトをロードすることで Web ページを生成する場合、先行するリクエストの処理に時間が掛かると後続の処理が滞ってしまう<sup>13</sup>。Web ブラウザから複数の TCP 接続を張ることによってある程度多重化を図ることは可能だが、本質的な解決とは言えず、リクエスト数を減らすなどプロトコル以外の部分を工夫する必要がある。

**HTTP/2** HTTP/2 では、HTTP/1.1 における上記の問題を解決するために、様々な高速化技術が取り入れられている。

まず、複数のリクエストを同時に処理するため、HTTP/2 では 1 つの接続上に仮想的な双方向シーケンスであるストリームを複数確立できる (ストリームの多重化)。各ストリームは独立しているため、レスポンスに時間が掛かるような場合でも、他のリクエストやレスポンスを処理できるようになった。ストリーム上では、データはフレームと呼ばれる単位でやりとりされ、クライアント側からストリームに対して優先度を付与することにより、表示に関係ないリソースの読み込みを後回しにする提案ができる。サーバ側からは、サーバプッシュ機能により、クライアント側からのリクエストを待つことなくコンテンツを送信することができる。

HTTP/1.1 では前述の通りテキストベースでやりとりが行われるが、HTTP/2 ではバイナリベースで行われる。HTTP ヘッダは、HPACK と呼ばれる方式で圧縮され、複数のリクエスト間ではヘッダの差分のみを送信する。HPACK は、頻出するヘッダ名と値を辞書として共有し、その他のヘッダは動的辞書として随時クライアント側で更新することにより、辞書のインデックス値を用いた効率的な圧縮を行うことができる。

HTTP/2 の効果を楽しむためには、サーバ側とクライアント側がともに HTTP/2 に対応している必要がある。ただし、非対応であっても、HTTP/1.1 を利用して通信自体は可能である。

<sup>13</sup>HTTP/1.1 における HoL (Head-of-Line) ブロッキング問題と呼ばれる。

HTTP/2 は、従来のバージョンと完全な互換性を保っているため、HTTP/2 に対応するクライアントは下記のように HTTP/1.1 と同じ方法で Web サーバに対して接続を行い、接続先が HTTP/2 に対応していることを確認してから HTTP/2 通信へと移行する。具体的には、Connection ヘッダ、Upgrade ヘッダ、HTTP2-Settings ヘッダをリクエストとして送信することでクライアント側が HTTP/2 に対応していることを伝え、HTTP/2 に対応しているサーバ側は 101 ステータスコードをレスポンスとして返すことにより、HTTP/2 接続へと移行する。

リクエスト例)

```
GET / HTTP/1.1
Host: portal.nagoya-u.ac.jp
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
HTTP2-Settings: <Base64 形式でエンコードされた設定情報>
```

レスポンス例)

```
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: h2c

<HTTP/2 データ ...>
```

暗号化通信である SSL/TLS プロトコルによって提供されるセキュアな接続の上で HTTP 通信を行うことを HTTPS (Hypertext Transfer Protocol Secure) と呼ぶ。ここ数年、Web サイト上のすべてのページを SSL/TLS (以下 TLS) に対応させる取り組みが進んでおり、セキュリティ強化に加えて、TLS が必須とされる HTTP/2 によるページ表示速度の向上などのメリットがある。

HTTPS では、ポート番号として 443 が使用される。HTTP/2 は、規格上 TLS による暗号化がなくても利用できることから、TLS で保護された HTTPS で HTTP/2 を利用する場合は h2 (HTTP/2 over TLS) という識別子を用い、暗号化されていない TCP 接続上で HTTP/2 を利用する場合は h2c (HTTP/2 over TCP) という識別子を用いて区別する。しかし、最近の Web ブラウザは h2c に未対応であることから、HTTP/2 を利用するためには HTTPS (TLS 1.2 以降) が必須となっている。

### 2.3.9 SMTP (Simple Mail Transfer Protocol)

SMTP サーバは、接続してきたクライアントからのリクエストにより、宛先のメールアドレスとメールデータ本体の配送を引き受ける。SMTP は様々な拡張を施された歴史を持っており、プロトコルを規定する RFC 文書も数多く存在する。しかし、HTTP の場合と異なりほとんどの拡張は旧来の規定と互換性を保ちながら、古い規定を破棄する形で提案されてきている。このため、規則の理解のためには、最も新しいもののうちいくつかを参照すれば充分である。具体的には RFC2821 を参照すればよい。

メールアドレスは次のような文字列で表される。

```
username@example.net
```

“@” (アットマーク) より前の部分をユーザ名、後の部分をメールドメインと呼ぶ。各メールドメインに対して、対応するメールボックスを管理するサーバが存在し、その IP アドレスないし FQDN が DNS の MX レコードとして管理されている。したがって、HTTP における URL の場合と同様にメールアドレスの一部から抽出されるメールドメインを利用して SMTP 通信の相手となるサーバを決定することができる。

SMTP サービスは TCP 通信を用いて提供されるので、telnet コマンドを用いて動作の確認を行うことができる。サーバで標準的に利用されるポート番号は 25 番なので、次のように実行すればよい。

```
% telnet サーバのホスト名または IP アドレス 25
```

基本的に、通信を確立しクライアントのリクエストに対して応答を送信すると、サーバ側から通信を終了させる。クライアント側から見た SMTP の接続後の手順は、まず、送信側クライアントのホスト名等の告知で始まる。サーバの了解を経て発信者のメールアドレスの通知、さらに了解を経て受信者のメールアドレスの通知と続く。受信者アドレスは複数指示する場合もあり、受信者に関して了解を得られたら、最後にメールのメッセージ本体を送信する。

telnet コマンドを利用したやり取りは HTTP とよく似ているが、SMTP ではクライアント側からの要求は、

コマンド	オプション
------	-------

の形式で出され、サーバはコマンド毎に

応答コード	応答メッセージ
-------	---------

の形式で応答する。また通信を終了するコマンドの要求がない限り接続は継続される。

### 2.3.10 POP (Post Office Protocol)

初期の POP は SMTP と関連して提案され、同様に様々な変遷を経て、現在では POP3 と呼ばれる改訂版にあたるプロトコルが利用されるようになった。POP3 自体にも様々な拡張が試みられているが、基本部分の詳細に関しては RFC1939 を参照すれば良い。

受信者がローカルなユーザであった場合、サーバに保存されたメッセージをユーザが利用できるようにする必要がある。保存したメッセージをユーザが操作するのに使われるのが POP である。

POP3 サービスは TCP 通信を用いて提供されるので、telnet コマンドを用いて動作確認を行うことができる。

```
% telnet [サーバのホスト名または IP アドレス] 110
```

POP も SMTP と同様にコマンド毎に応答が行われ、QUIT コマンドで終了するプロトコルである。SMTP と異なるのは、ユーザ別に蓄積されたメッセージの操作を行うので、ユーザの認証が行われるという点と、データへのアクセスに対して排他制御が行われる点である。基本的な通信の手順は次の通りである。

1. 通信確立後、まずユーザ名とパスワードを用いて認証を行う。
2. メールボックスに保存されているメッセージの有無等を確認する、
3. 必要なメッセージを指定して読み込む。
4. 不要なメッセージを指定してメールボックスから削除する。
5. クライアント側から通信を終了させる。

### 2.3.11 FTP (File Transfer Protocol)

FTP はネットワークに接続された複数の計算機の間でファイルを転送するための仕組みであり、その詳細は RFC959 で規定されている。サーバ側で利用される標準のポート番号は、コントロールコネクションが 21 番、データコネクションが 20 番である。

FTP もコマンドによるリクエストとコマンド毎の応答に関する規則で構成される点で、部分的には SMTP/POP と共通している。一方で、コマンドのやりとりのための通信とは別に、データのやりとりをするための通信を用いる点で異なる。FTP というプロトコルはコマンドの送受信のためのプロトコルとデータの送受信のためのプロトコルの 2 つの規則からなるということである。

**コントロールコネクションとデータコネクション** 前述の通り、FTP では役割の異なる二つの通信を利用する。これらのうち、コマンドによるリクエストや対応する応答をやりとりするための通信をコントロールコネクション、データのやりとりのための通信をデータコネクションと呼ぶ。ここでは、2 つの通信のうちコントロールコネクションの両端をクライアントとサーバと考えて説明を行う。

クライアントのリクエストを受けてサーバはファイルの送受信を行なう。コントロールコネクションにおいてクライアントが指示するのは、送受信対象となるサーバのファイルの特定と、ファイルの内容となるデータを送受信する通信の方法である。コントロールコネクションにおける基本的な通信の手順は次のようになる。

1. 通信確立後、まずユーザ名とパスワードを用いて認証を行なう。
2. データコネクションの通信方法と通信相手を指定する。データコネクションの指定方法には2通りある。
3. ファイルの操作を行う。基本的な操作は次の通りである。
  - 操作対象のディレクトリを指定する。
  - データコネクションから読み込んだデータをファイルに保存する。
  - ファイルの内容をデータコネクションを通じて送信する。
  - ファイルを削除する。
4. クライアント側から通信を終了させる。

コントロールコネクションにはTCP通信を用いるので、telnet コマンドを用いて動作の確認を行うことができる。サーバで標準的に利用されるポート番号は21番なので次のように実行すればよい。

```
% telnet サーバのホスト名またはIP アドレス 21
```

ファイル操作のコマンドの詳細はRFC959を参照。データコネクションの設定に関しては、次の2通りの方法が利用できる。一つ目がPORTモードあるいはアクティブモードと呼ばれる方法で、

```
PORT IP アドレスとポート番号
```

というコマンドでデータコネクションの通信相手を指定する。このとき、IP アドレスとポート番号の指定にはカンマ区切りの6つの十進数を用いる。先頭の4つがIPアドレスの4オクテットに対応し、残る2つがポート番号の上位バイトと下位バイトに対応する。二つ目がパッシブモードと呼ばれる方法で、

```
PASV
```

というコマンドに対してサーバが応答してくるポート番号を用いる。

RETRやSTOR等のファイル転送を実行するコマンドがクライアントから送信されたときのサーバ動作は、そのときに設定されているモードによって異なる。PORTモードが設定されていれば、サーバはクライアントの指定する相手に通信を開始しデータの送受信を行う。PASVモードが設定されていた場合、サーバは自ら指定したポートで通信を待ち、外部からの通信を受けてデータの送受信を行う。PASVモードであれば、サーバの指定するポートへの通信を行うことでtelnetコマンドによりデータコネクションの動作を確認することが可能であるが、PORTモードのためには通信を待ち受けるサーバプログラムを別に用意する必要がある。つまり、PORTモードではコントロールコネクションのサーバがデータコネクションのクライアントに、PASVモードではサーバに相当することになる。

### 2.3.12 ICMP (Internet Control Message Protocol)

プロトコルの詳細はRFC792を参照。ICMPは、IPのエラーメッセージや制御メッセージを転送するプロトコルであり、TCP/IPで接続されたコンピュータやネットワーク機器間で、互いの状態を確認するために用いられる。ユーザやアプリケーション自身が明示的にこのプロトコルを利用したり、送受信したりすることは少なく、唯一、ネットワーク診断プログラムのpingコマンドで利用するくらいである。しかし、ICMPはTCP/IPネットワークが円滑に稼動するために欠かせない重要なプロトコルである。

ping<sup>14</sup>は、TCP/IPネットワークにおいて、ノードの到達性を確認するツールである。pingは、ICMPの“echo request”パケットを対象ノードに投げ、対象ノードから“echo reply”が返ってくることによって到達性を確認している。

<sup>14</sup>「ピング」とか「ピン」と呼ぶ。プログラムの挙動が潜水艦などで使われるアクティブソナーの発する音波の挙動と似ていることからこの名が付けられた。



ping では、リプライが返ってくるまでの時間や応答率から、対象ノード間のラウンドトリップタイムやパケットロス率を求めることができ、これらは対象ノード間の回線状況を知るための重要な情報である。

Linux 端末から icecs.ice.nuie.nagoya-u.ac.jp に対して ping を実行した場合の出力例<sup>15</sup>を以下に示す。

```
$ ping icecs.ice.nuie.nagoya-u.ac.jp
PING icecs.ice.nuie.nagoya-u.ac.jp (10.10.1.7) 56(84) bytes of data.
64 bytes from www (10.10.1.7): icmp_seq=1 ttl=64 time=0.206 ms
64 bytes from www (10.10.1.7): icmp_seq=2 ttl=64 time=0.241 ms
64 bytes from www (10.10.1.7): icmp_seq=3 ttl=64 time=0.202 ms
64 bytes from www (10.10.1.7): icmp_seq=4 ttl=64 time=0.168 ms
64 bytes from www (10.10.1.7): icmp_seq=5 ttl=64 time=0.127 ms

--- icecs.ice.nuie.nagoya-u.ac.jp ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.127/0.188/0.241/0.042 ms
```

### 2.3.13 ARP (Address Resolution Protocol) / RARP (Reverse Address Resolution Protocol)

プロトコルの詳細は RFC826 (ARP), RFC903 (RARP) を参照されたい。ARP は、Ethernet 環境において IP アドレスから MAC アドレスを得るために用いられるプロトコルである。TCP/IP では、IP アドレスで通信相手を持定し、その IP アドレスに対して IP パケットを送信する。相手のコンピュータやルータに対して IP パケットを送信するためには、Ethernet 環境の場合、IP アドレスではなく MAC アドレスを利用する。そのため、IP アドレスと MAC アドレスの対応表が必要となり、ARP はそれを作成するためのプロトコルである。

MAC アドレスを調べる場合、自ノードの IP アドレスと MAC アドレス、そして対象ノードの IP アドレスの情報を要求パケットに格納して、Ethernet ネットワークにブロードキャストする (ARP リクエスト)。要求パケットを受け取ったネットワーク上の各ノードは、自分の IP アドレスと同一であれば、自分の MAC アドレスを送信元に返答する (ARP リプライ)。ARP によって調査された IP アドレスと MAC アドレスの対応表 (ARP キャッシュテーブル) は、一定時間該当ノード上で保持される。

RARP は、ネットワーク機器の MAC アドレスから IP アドレスを取得するためのプロトコルである。ネットワーク機器は、自分の MAC アドレスをブロードキャストし (RARP リクエスト)、それに対して RARP サーバが応答して IP アドレスを配布する。主に、DHCP や BOOTP において、IP アドレスを取得するために RARP が用いられる。

arp コマンドにより、Linux で ARP テーブルの内容を表示する例を以下に示す。

```
$ /sbin/arp
```

Address	HWtype	HWaddress	Flags Mask	Iface
ssh	ether	00:1D:60:95:09:7D	C	eth0
icemgr	ether	00:0A:E4:83:0B:B2	C	eth0
icefs	ether	02:A0:98:0A:49:11	C	eth0

## 2.4 ルータと静的ルーティング制御

2.1.6 項の「c) ルーティング (通信経路の選択)」で述べたように、ルーティングには大きく分けてスタティック (静的) ルーティングとダイナミック (動的) ルーティングの 2 種類がある。

スタティックルートとは、管理者が宛先のネットワークへの最適な経路を手動で設定したものである。ダイナミックルーティングと異なり、スタティックルートの情報は他のルータに通知されることはなく、ネットワークの

<sup>15</sup>Linux の場合、Ctrl+c で止めるまで ping を打ち続ける。

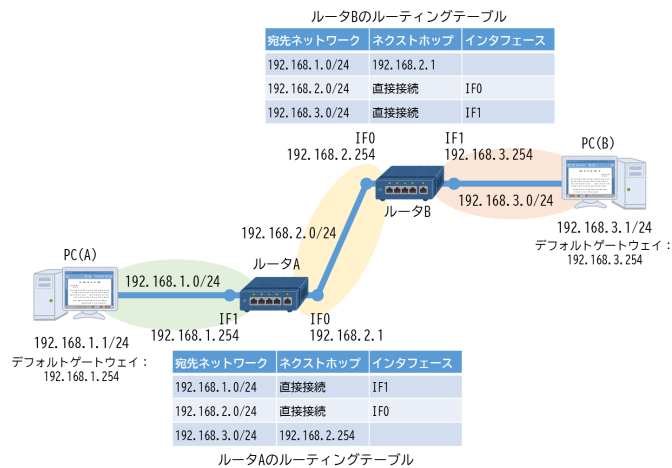


図 9: スタティックルートの設定例 1

状態に変化が発生して、有効な別の経路が存在しても自動的にその経路に切り替わることはない。ルータでスタティックルートを使用したルーティングのことを、スタティックルーティングと呼ぶ。

スタティックルーティングのルーティングテーブル（経路表）で重要な情報は下記の 3 つである。

- 宛先ネットワーク：ルータが管理している宛先ネットワークのネットワークアドレスとネットマスク
- ネクストホップ：宛先ネットワークに到達するために、IP パケットを転送すべき隣接ルータのアドレス
- 出力インタフェース：ネクストホップアドレスに IP パケットを転送するために使うインタフェース

#### 2.4.1 スタティックルートの設定

模擬的に 2 台の PC の間に 2 つのルータを挟んで接続した場合のスタティックルートの設定例を図 9 に示す。ルータ A には、192.168.1.0/24 と 192.168.2.0/24 の 2 つのネットワークがそれぞれ IF1 と IF0 のインタフェースに直接接続されている。ルータ B には、192.168.2.0/24 と 192.168.3.0/24 の 2 つのネットワークがそれぞれ IF0 と IF1 のインタフェースに直接接続されている。PC(A) から PC(B)、PC(B) から PC(A) へのパケットを届けるために、ルータ A には 192.168.3.0/24 を宛先とするパケットを隣接するルータ B の 192.168.2.254 に転送し、ルータ B には 192.168.1.0/24 を宛先とするパケットを隣接するルータ A の 192.168.2.1 に転送するよう設定している。

なお、PC(A) と PC(B) はルーティング機能を持たないため、自分の知らないネットワークへの通信はすべて指定されたネクストホップアドレスに転送する。このアドレスをデフォルトゲートウェイと呼ぶ。

#### 2.4.2 デフォルトルートの設定

自宅や研究室のように、無線 LAN ルータやルータ機能を持つスイッチ、ルータ機能を持たせたサーバなどを介してインターネットに接続した場合のスタティックルートの設定例を図 10 に示す<sup>16</sup>。

図 10 左では、2 台の PC がルータを介してインターネットに接続されている。ルータには、192.168.1.0/24 と 192.168.100.0/24 の 2 つのネットワークがそれぞれ IF1 と IF0 のインタフェースに直接接続されている。2 台の PC とインターネット間の通信を可能にするためには、インターネット上の宛先ネットワークを 1 行ずつルーティングテーブルに記載しなければならない。

しかし、通信したい宛先ネットワークをスタティックルートとしてすべて設定するのは非常に手間であるため、ルーティングテーブル上に一致するエントリがない場合に、指定したネクストホップアドレスにパケットを転送してくれる特別な経路が用意されている。これをデフォルトルートと呼び「0.0.0.0/0」で表す。

<sup>16</sup>説明の都合上省略しているが、インターネットとルータの間には、自宅の場合、ADSL 回線で送られてくるアナログ信号をデジタル信号に変換するモデムや、光信号をデジタル信号に変換する光回線終端装置が入っており、研究室の場合は、適切にルーティングの設定が行われている上位のルータ（L3 スイッチ）などが入っている。

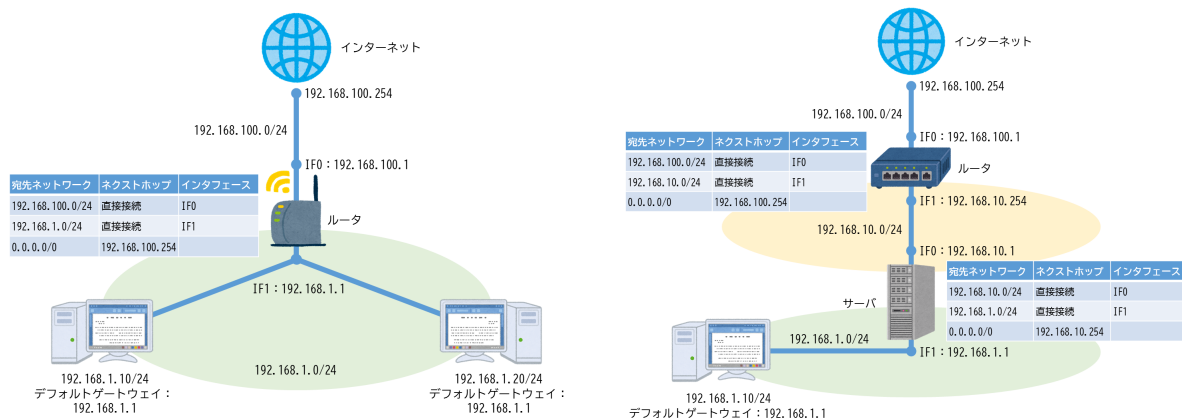


図 10: スタティックルートの設定例 2

図 10 左の場合、インターネット上の宛先ネットワークをデフォルトルートとし、0.0.0.0/0 を宛先とするパケットを 192.168.100.254<sup>17</sup>に転送するよう設定している。

図 10 右では、PC がルータ機能を持たせたサーバとルータを介してインターネットに接続されている。サーバには、192.168.1.0/24 と 192.168.10.0/24 の 2 つのネットワークがそれぞれ IF1 と IF0 のインタフェースに直接接続されており、ルータには、192.168.10.0/24 と 192.168.100.0/24 の 2 つのネットワークがそれぞれ IF1 と IF0 のインタフェースに直接接続されている。

PC とインターネット間の通信を可能にするために、サーバのルーティングテーブルには、インターネット上の宛先ネットワークをデフォルトルートとし、0.0.0.0/0 を宛先とするパケットを隣接するルータの 192.168.10.254 に転送するよう設定している。ルータのルーティングテーブルも同じく 0.0.0.0/0 を宛先とするパケットをインターネット側のアドレス 192.168.100.254 に転送するよう設定している。

ルーティング機能を持たない PC や L2 スイッチの場合、すべての宛先ネットワークをデフォルトルートとし、0.0.0.0/0 を宛先とするパケットをデフォルトゲートウェイとして指定したアドレス（図 10 の場合、192.168.1.1）に転送するよう設定されている。

<sup>17</sup> 自宅の場合には、ISP（インターネットサービスプロバイダ）が指定するアドレスである。

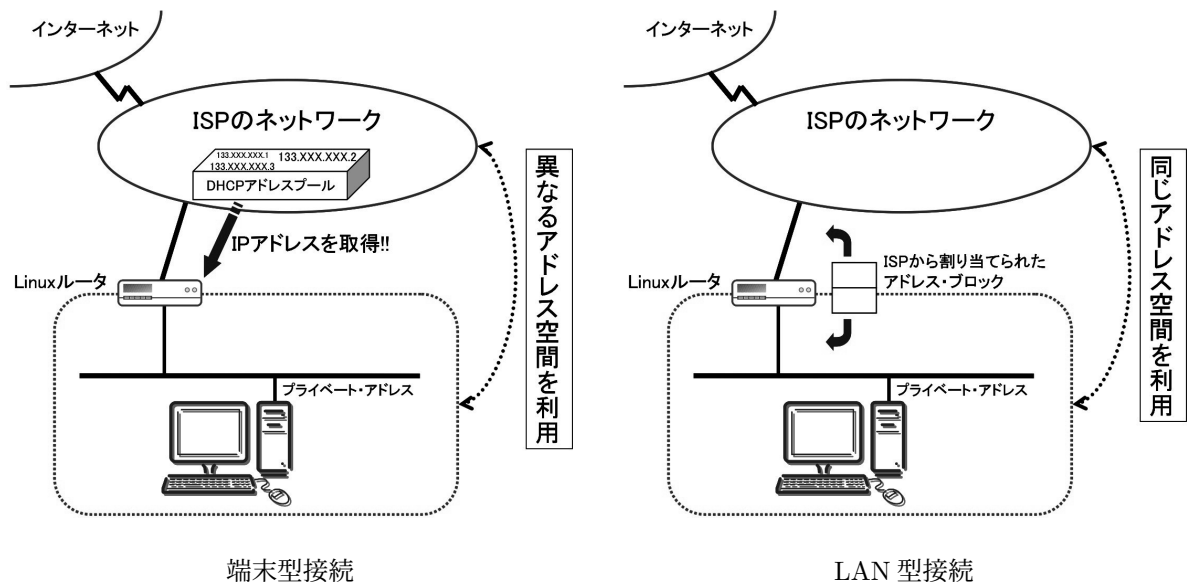


図 11: 端末型接続と LAN 型接続

### 3 情報セキュリティ対策

CATV や ADSL を利用してインターネットに接続している場合には、ケーブルモデムや ADSL モデムを使って接続会社の回線に接続する。このとき、モデムには一般には“データの変調と復調”の機能しかなく、ルータのようにフィルタリングなどの機能を持つ製品は現状では稀である。そのため、これらの機器を用いてインターネットに接続している場合は、ファイアウォールを構築するためのルータを別途用意する必要がある。実際には、わざわざルータを用意せずとも、部屋の片隅に転がっている計算機にフリーの PC UNIX を入れてファイアウォールを構築するだけでもそれなりのセキュリティ対策ができる。本節では、Linux の“IP パケット・フィルタリング機能”を利用してファイアウォールを構築する方法を以下の順序で説明する。

1. 端末型接続と LAN 型接続
2. IP マスカレード
3. netfilter
4. iptables によるパケットフィルタリングとアドレス変換
5. フィルタリング・ルールの設定
6. アドレス変換の設定
7. ログ管理

詳細については文献 [13] を参照されたい。

#### 3.1 端末型接続と LAN 型接続

インターネットへの接続（プロバイダなどが提供するサービス）には端末型接続と LAN 型接続がある。ルータの代わりに Linux マシンを使う場合には、それぞれ図 11 のようなイメージになる。

**端末型接続：** 1つの IP アドレスが静的に割り当てられるサービス、あるいは1つ以上のアドレスが動的に割り当てられるサービス。IP アドレスを“動的”に取得する手段としては、PPP (PPPoE: PPP over Ethernet) や DHCP などがある。

内部ネットワークには独自のプライベート IP アドレスを割り当て、外部と通信するにはルータ上で IP マスカレードによる変換を行う。

**LAN 型接続：** 1つのブロック、つまり連続した複数の IP アドレスが静的に割り当てられるサービス。アドレス変換 (NAT) や IP マスカレードなどによるアドレス変換は行わず、内部ネットワークには外部と同じ空間の IP アドレスを割り当てる。

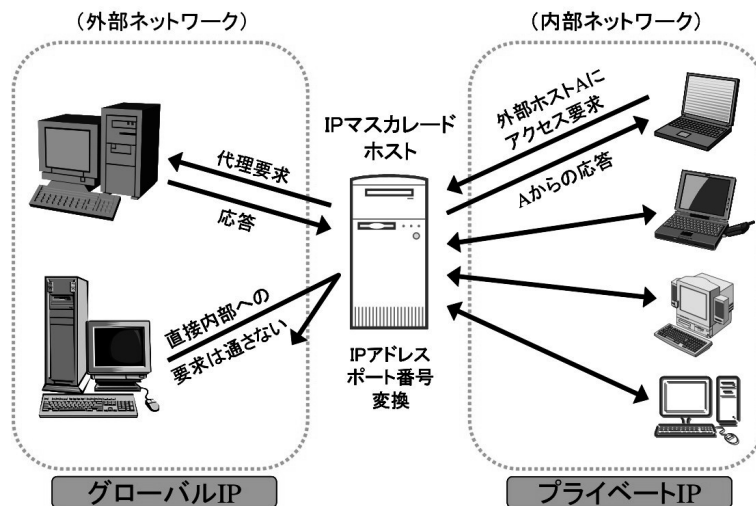


図 12: IP マスカレードとパケットフィルタリング

本節では、端末型接続を前提として説明する。

### 3.2 IP マスカレード

IP マスカレード (IP Masquerade) とは、IP アドレスとポート番号を変換することで、1つのグローバルアドレスを利用して複数のローカルコンピュータが外部 (インターネット) に接続することのできる機能である。

この機能を持つ Linux ホスト 1 台がインターネットに接続されグローバルアドレスを取得していれば、そのホストに接続している任意台数の内部ネットワークに設置されたコンピュータは、プライベート IP アドレスを割り当てられている状態で、インターネット側とやりとりができる。

NAT (Network Address Translation) は、特定のグローバルアドレスを特定のプライベートアドレスに 1 対 1 で変換する仕組みであり、IP マスカレードも広義の NAT の 1 つである。

IP マスカレードの機能を持つホストに接続している内部ネットワークのコンピュータは、マスカレードの背後に隠れてインターネットにアクセスするため、外部からはインターネットに接続しているゲートウェイとなるホストだけしか見えない。すなわち、内部から外部宛てに出した要求の応答は通すが、これ以外の外部からのアクセスを制限することができ、内部のセキュリティを高めることができる。

### 3.3 netfilter

Linux カーネルは、バージョン 2.4 でパケットフィルタリングの仕組みを大きく変えた。そこで、本小節ではまずバージョン 2.4 から採り入れられた netfilter [12] という仕組みと、その上に実装された iptables について説明する。

#### 3.3.1 netfilter とは

カーネル 2.2 には、ipchains というパケット・フィルタリングの仕組みと、パケットの IP アドレスを変換する IP マスカレードが実装されていた。これらは、どちらも IP のパケットを処理するため、プログラムは同じ IP 層に組み込まれている。しかし、もともとは個別に開発されたプログラムをカーネルに組み込んだ結果、IP 層の部分が“読みにくく、拡張しにくい”コードになってしまっていた。そのため、どの順番でどの処理が行われるのかを理解していないと、適切なフィルタリング・ルールが書けなくなっていた。すなわち、ipchains は“見通しの悪いシステム”であった。

これらの点を考慮し、“すっきりしていて、かつ拡張しやすいシステム”を目指して netfilter と呼ばれるフレームワークが開発された。そして、その上に実装されたフィルタリング・システムが iptables である。

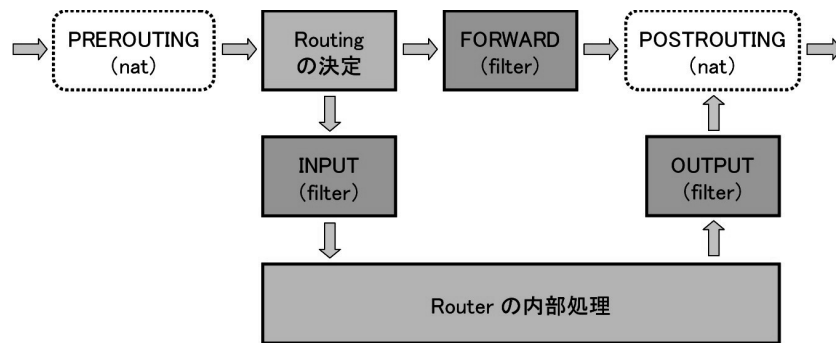


図 13: iptables の処理の流れ

カーネル 2.4 の iptables では、パケットの処理は図 13 のように行われる。  
この図からも分かる通り、パケットの処理は以下のように単純化されている：

- INPUT：受信したパケットに対するチェーン
- OUTPUT：送信するパケットに対するチェーン
- FORWARD：転送する (対象サーバを経由する) パケットに対するチェーン
- PREROUTING：受信時にパケットのアドレス変換を行うチェーン
- POSTROUTING：送信時にパケットのアドレス変換を行うチェーン

さらに、転送するパケットをチェックするとき、iptables では FORWARD チェーンしか通らないのが大きな特徴である。

これらの変更によって、次のような利点が得られる。

- 柔軟なフィルタが書きやすい。
- パケット転送時のパフォーマンスが向上する。

netfilter の利点はほかにもある。構成が単純になったため、処理ルーチンのモジュール化が容易となり、モジュールの開発や追加が簡単に行えるようになった。実際、カーネル 2.4 にはさまざまなモジュールが用意されている。これらをカーネルに組み込むと、以下の機能が使えるようになる。

**ステートフル・インスペクション：** パケット・フィルタリングを行うシステムが、転送するパケットの内容を解析して“コネクション”を解釈し、コネクションに必要なフィルタリング・ルールを動的に追加/削除してくれる機能である。単純なパケットフィルタリングでは、コネクションの“戻りパケット”であるかのように偽造されたパケットの流入を防ぐのは容易ではない。しかし、この機能を利用すれば、この種の偽造パケットの流入が防げる。

**レート・フィルタリング：** 単位時間当たりのパケット数を制限する仕組みである。

**パケットオーナー：** パケットを送信したプロセス ID やユーザ ID を特定する仕組みである。これが利用できるのは OUTPUT チェーンだけである。

### 3.3.2 netfilter の仕組み

カーネルの IP 層では、上位層または下位層から受信したデータを適切な宛先に送信する処理（ルーティング, routing）が行われる。例えば、受信したパケットが自分宛ならば上位層に渡し、そうでなければ適切なネットワークへ転送する。パケットを送信する場合も、同じく適切なネットワークを選んで送り出す。netfilter における IP 層での処理を図 14 に示す。

netfilter は、図 14 の 1～5 の部分でそれぞれ“フック関数”を呼び出す仕組みを提供している。

**フック**とは、あらかじめ決められた箇所に関数を登録しておく、そこに制御が到達したとき、自動的にその関数が呼び出されることをいう。番号だけでは憶えにくいので、図 14 の 1～5 には、それぞれ次のような名前が付けられている。



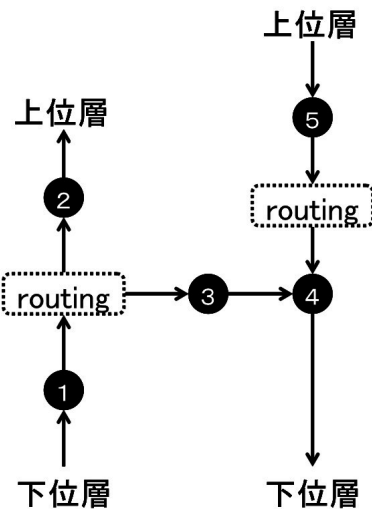


図 14: netfilter のフック

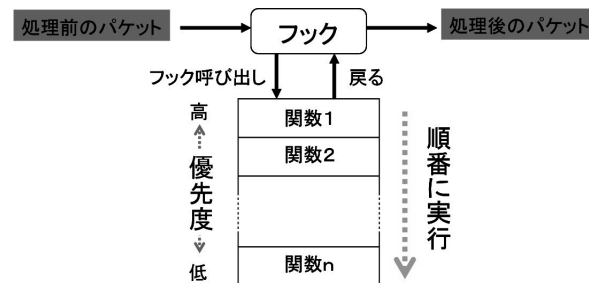


図 15: フックからの関数の呼び出し

- 1 NF\_IP\_PRE\_ROUTING
- 2 NF\_IP\_LOCAL\_IN
- 3 NF\_IP\_FORWARD
- 4 NF\_IP\_POST\_ROUTING
- 5 NF\_IP\_LOCAL\_OUT

これらの箇所には、関数を登録できるリストが用意されており、それぞれの箇所に制御が到達したとき、関数が優先度順に呼び出される仕組みになっている（図 15）。

それぞれの箇所に関数を登録するには、netfilter が提供するインタフェースを使用する。また、関数の優先度は、フックへの登録時に指定する。さらに、フックに登録する関数の引数の型と順番、関数の戻り値も netfilter のフレームワークの中で定義されている。ただし、決められているのはこれらの規則に従うことだけで、関数内部で行われる処理については特に決まりはない。すなわち、パケットをどうしようが netfilter としては関知しないので、始点や終点の IP アドレスを書き換えたり、パケットを破棄することも可能である（このときは、以降の処理が中断される）。

以上の説明より、netfilter とは“フックを登録し、呼び出す仕組み”といえる。

### 3.4 iptables

**iptables** は、netfilter のフレームワークの上に実装されたパケット・フィルタリングおよび NAT システムである。

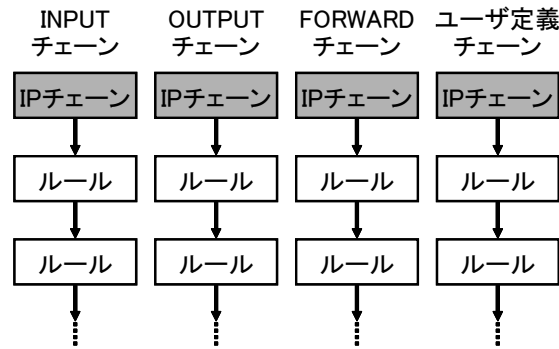


図 16: IP チェーン概念

### 3.4.1 パケット・フィルタリング

iptables では、パケット・フィルタリングの処理を行うためにチェーンが実装されている。チェーンは Linux で IP フィルタリングを行う際に、非常に重要なものである。まずはチェーンについて簡単に説明する。

**チェーン** (chain) とはフィルタリング・ルールの一覧である。Linux カーネルの中では、すべてのフィルタリング・ルールは必ずどれか 1 つのチェーンに属し、各チェーンはリスト構造で管理されている。

カーネルは、IP パケットを受信すると、INPUT チェーンを上から順にチェックしていき、最初にマッチしたフィルタリング・ルールに従ってそのパケットを処理する。同様に、IP パケットを送信するときは OUTPUT チェーン、転送するときには FORWARD チェーンを調べる。通常のファイアウォールを構築する場合であれば、INPUT と OUTPUT と FORWARD のルールがあり、動作として ACCEPT と DROP の 2 つが指定できれば十分である。

INPUT, OUTPUT, FORWARD の 3 つのチェーンはカーネルの中に必ずデフォルトで存在し、**ビルトイン・チェーン**と呼ばれる。さらに、ユーザが新たに作成するチェーンを**ユーザ定義チェーン**と呼ぶ。ビルトイン・チェーンとユーザ定義チェーンの違いは、参照される順番である。ビルトイン・チェーンと同様に、ユーザ定義チェーンにはフィルタリング・ルールを登録し、それらのルールは図 16 に示されるようにリストとして管理される。

パケットがルールにマッチしたときの動作には、ACCEPT (許可) と DROP (破棄) があるが、ユーザ定義チェーンの名前を指定することもできる。例えば、パケットがフィルタリング・ルールにマッチし、そのルールの動作としてチェーンの名前が指定されていたとすると、カーネルは指定されたチェーンのチェックを始める。もし、指定されたチェーンでマッチするルールがなければ、再び元のチェーンに復帰する。

iptables のチェーンは、netfilter の次の位置にフックをかけるように実装されている。

```

NF_IP_LOCAL_IN    <== INPUT チェーン
NF_IP_FORWARD     <== FORWARD チェーン
NF_IP_LOCAL_OUT   <== OUTPUT チェーン

```

iptables では、INPUT, FORWARD, OUTPUT の各チェーンの設定は他のチェーンに影響を及ぼさない。そのため、各チェーンの設計が簡単になる。

チェーンのルールの登録や変更は、iptables コマンドを用いて行う。また、登録されたルールはカーネルの **filter テーブル** という領域に登録される。

### 3.4.2 アドレス変換

iptables には、パケットの IP アドレスを変換する NAT だけでなく、IP マスカレードとポート・フォワーディングの機能も実装されている。これらの NAT に関連する処理は、次のフックで処理される。

```

NF_IP_PRE_ROUTING <== PREROUTING チェーン (終点 IP アドレス変換)
NF_IP_POST_ROUTING <== POSTROUTING チェーン (始点 IP アドレス変換)

```

NAT の変換ルールを登録したり、変更したりするには、フィルタリングと同様に iptables コマンドを使用する。また、その結果はカーネルの **nat テーブル** という領域に登録される。



### 3.5 フィルタリング・ルールの設定

前述したように、iptables のフィルタリング・ルールを追加、編集するときは **iptables コマンド**を使う。本節では、iptables コマンドのオプションと使用方法を説明する。iptables コマンドを使用するには、モジュール ip\_tables がカーネルに組み込まれていなければならない。組み込まれていない場合は次のコマンドを実行し、カーネルにモジュールを読み込む。

```
# rmmod ipchains
# modprobe ip_tables
```

iptables コマンドの書式は次のようになる。

```
iptables -{ A | I | D } チェーン名 [ルール番号]
        [-i !] インタフェース名[+] [-o !] インタフェース名 [-p !] プロトコル
        [!] [-f] [-s !] 始点 IP アドレス [-d !] 終点 IP アドレス
        -j ターゲット
```

#### 3.5.1 フィルタの登録と削除

iptables コマンドでフィルタを登録・削除する場合には、以下のオプションを指定する。

- A チェーン名： フィルタの登録。指定されたチェーンの末尾にルールを追加する。
- I チェーン名 [ルール番号]： フィルタの登録。指定されたチェーンの先頭（ルール番号の位置）にルールを追加する。記述されたルールが先頭から何番目にあたるかは、(後述の) オプション-L により調べることができる。
- D チェーン名 [ルール番号]： フィルタの削除。指定されたチェーンからのルールを削除する。削除するルールは、次の 2 つのうちのどちらかの方法で指定する。
  - 登録したときと同じ内容のルールを書く。
  - チェーンの何番目かをルール番号で指定する（この場合はルールの内容を書く必要はない）。

**チェーン名**： ルールを登録・削除するチェーンの名前は下記の中から選ぶ。

- INPUT
- OUTPUT
- FORWARD
- ユーザ定義チェーンの名前（チェーンの作成方法は、後で説明する。）

- i !] インタフェース名[+]： 受信ネットワーク・インタフェース。パケットを受信したインタフェースの名前を指定する。このオプションは、INPUT チェーンと FORWARD チェーンと PREROUTING チェーンのルールに対して指定することができる。インタフェース名の後ろに+を付けると、その名前で始まるすべてのインタフェースにマッチする。例えば、ppp+は ppp0, ppp1, ... にマッチする。
- o !] インタフェース名： 送信ネットワーク・インタフェース。パケットを送信するインタフェースの名前を指定する。このオプションは、OUTPUT チェーンと FORWARD チェーンと POSTROUTING チェーンのルールに対して指定することができる。
- p !] プロトコル： プロトコル。フィルタリングの対象とするパケットの種類を指定する。プロトコルには、プロトコルの番号（数字）または文字列（udp, tcp, icmp）を記述する。オプション all を指定するか、もしくはこのオプションを省略した場合は、すべてのプロトコルにマッチする。プロトコルの前に!を付けると、指定したものを除くすべてのプロトコルをフィルタリングの対象とする。

[!] `-f` : フラグメント化されたパケットの 2 番目以降のフラグメントにマッチする。オプションの前に!を付けると、結果が反転、すなわち、フラグメント化されていないパケットと、フラグメントの先頭パケットにマッチする。

IP パケットの大きさがネットワークの MTU (Maximum Transfer Unit) よりも大きいと、小さく分割 (フラグメント化) される。分割されたパケットは受信側で再構成される。しかし、フラグメント化パケットを再構成する方法には規格がなく、実装 (OS) ごとに異なる。そのため、ある実装をターゲットとしたとき、特別な順番でデータを配置したフラグメント化パケットを送ると、途中にあるパケット・フィルタリング型のファイアウォールをすり抜けてコネクションを確立できる場合がある。また、フラグメント化パケットの再構成ルーチンには、過去にいくつかの実装でバグが発見されている。そのバグの影響は深刻で、カーネルの停止やパニックが発生する。また、これを悪用して“フラグメント化されているかのように見える”パケットを送信する攻撃も存在する。

このオプションは、このようなフラグメントを悪用する攻撃を防ぐときに使用する。ただし、カーネル・パラメータの `net.ipv4.ip_always_defrag` を有効 (1) にしていると、INPUT チェーンをチェックする前にカーネルがパケットを再構成するので、このオプションは常に“偽”となる。セキュリティの面ではよさそうに感じるが、これまでに上記の“カーネルがパニックするバグ”が Linux で何度も発見されているという事実を考えると、安易に有効にするのも怖い。しかし、内側の計算機がパニックするくらいなら、Linux ファイアウォールが犠牲になった方がよいと考えるなら、有効にしてもよいかもしれない。

-s [!] 始点 IP アドレス : 始点 IP アドレスを指定する。ネットワーク・アドレスを指定するときは、次のように / に続けてネットマスク長、またはマスクアドレスを記述する。

```
133.6.202.0/24
133.6.202.0/255.255.255.0
```

特定の計算機の IP アドレスを指定するときは、アドレスをそのまま記述する。

```
133.6.202.1
```

すべての IP アドレスを指定する場合は、`0.0.0.0/0` と記述する。また、省略した場合はすべてのアドレスにマッチする。

アドレスの前に!を付けると、指定したもの以外のすべてのアドレスをフィルタリングの対象とする。

-d [!] 終点 IP アドレス : 終点 IP アドレスを指定する。IP アドレスの記述方法は始点 IP アドレスと同様である。

-j ターゲット : 以下の 5 つの中から、いずれか 1 つを選ぶ。なお、大文字と小文字は区別される。

- `ACCEPT` : 許可する。
- `DROP` : 破棄する。
- `QUEUE` : 他の計算機宛のパケットを横取りし、ルールに指定されたポートにリダイレクトする。リダイレクトされたパケットは、ローカルのソケットから取り出すことができる。
- `RETURN` : このターゲットはチェーンの種類によって動作が異なる。ユーザ定義チェーンでは、そのチェーンを呼んだ次のルールに戻る。INPUT, OUTPUT, FORWARD チェーンの場合、その動作は後述のポリシーに応じて決定される。
- ユーザ定義チェーンの名前 : ユーザ定義チェーンのチェックを始める。ユーザ定義チェーンにマッチするルールがあり、そのターゲットが `RETURN` であった場合、またはどのルールにもマッチしなかった場合には、ルールの評価のポインタがもとのルールに戻る。

-j というオプションを省略すると、評価ポインタはそのルールを素通りして次のルールに移動する。ただし、ルールのチェック自体は行われ、マッチしたルールのカウンタが増加する。カウンタの参照方法は、後述の“フィルタの表示”で述べる。

上記のように、標準の iptables コマンドで指定できるマッチング・ルールはプロトコルの種類とアドレス、ネットワーク・インタフェース、フラグメント化されたパケットだけである。これでは、セキュリティを目的としたフィルタリングには明らかに機能不足である。

iptables コマンドは、モジュールを動的にロードして機能を拡張し、ルールをさらに細かく指定できるように作られている（そのための機能を実装したカーネル・モジュールも、自動的にカーネルに読み込まれる）。動的にロードするモジュールは `-m` オプションで指定し、そのファイルは `/lib/iptables` に置かれている。ただし、`-p` オプションでプロトコルを指定した場合には、対応するモジュールが自動的に読み込まれる。例えば、`-p tcp` と指定すると、`-m tcp` が自動的に補われる。さらに、iptables ではターゲットについても機能を拡張することができる。例えば、パケットのログを記録するターゲットなどもモジュールとして実装されている。

各モジュールで指定可能なオプションの種類と書式を知るには、次のように `-m`（または `-j` と `-h` オプションの両方を指定して実行する。

```
iptables -m モジュール名 -h
iptables -j モジュール名 -h
```

### TCP マッチング拡張モジュール：`-m tcp`

TCP のパケットのマッチング・ルールを指定するためのモジュールである。

`--sport [!] 始点ポート番号`：始点ポート番号を数字、または `/etc/services` に登録されている名前（文字列）で指定する。連続したポート番号を指定する場合は、次のように開始番号と終了番号（または、それぞれに対応する名前）を `:` で区切って記述する。

```
137:139
netbios-ns:netbios:ssn
```

最初のポート番号を省略すると 0 が補われる（ポート 0 は不正な値であるが、そのようなパケットが送られてくることがある）。逆に、2 番目のポート番号を省略すると 65535 が補われる。

ポート番号の前に `!` を付けると、指定したものを除くすべてのポートをフィルタリングの対象にする。

`--dport [!] 終点ポート番号`：終点のポート番号を指定する。記述の方法は、`--sport` と同様である。

`--tcp-flags [!] マスク 比較条件`：TCP のフラグを指定する。引数の マスク と 比較条件 には次の“フラグの種類”を記述し、複数指定する場合は、`:` で区切る。

```
SYN  ACK  FIN  RST  URG  PSH
ALL  NONE
```

ALL と NONE は特別な意味を持つ。ALL は SYN から PSF までのすべてのフラグがセットされていることを表し、NONE はすべてクリアにされていることを表す。

マッチング処理では、パケットに記録されたフラグを マスク でマスクし、その結果が 比較条件 と一致すればマッチしたと判断される。

[!] `--syn`：SYN ビット。次の 3 つの条件をすべて満たす TCP のパケットにマッチし、`--tcp-flags SYN,RST,ACK SYN` と指定した場合と同様の結果になる。

1. SYN ビットがセットされている。
2. RST ビットがクリアされている。
3. ACK ビットがクリアされている。

このオプションは、TCP 3 way handshake の 1 番目のパケットをマッチさせるときに使用する。

オプションの前に `!` を付けると結果が反転するので、3 way handshake の 2 番目以降のパケットにマッチさせたいときに使用できる。ただし、その場合には、異常なパケットが送られてきたときにもマッチしてしまうことに注意しなければならない。

## UDP マッチング拡張モジュール：-m udp

UDP のパケットのマッチング・ルールを指定するためのモジュールである。

--sport [!] 始点ポート番号： 始点のポート番号を指定する。記述の方法は TCP の --sport と同様である。

--dport [!] 終点ポート番号： 終点ポート番号を指定する。記述の方法は TCP の --dport と同様である。

## ICMP マッチング拡張モジュール：-m icmp

ICMP のパケットのマッチング・ルールを指定するためのモジュールである。

--icmp-type [!] タイプ名 [コード]： ICMP のタイプとコードを指定する。

## MAC アドレス・マッチング拡張モジュール：-m mac

パケットの送信元 MAC アドレスを指定するためのモジュールである。

--mac-source [!] 始点 MAC アドレス： MAC アドレスを “xx:xx:xx:xx:xx:xx” という書式で指定する。このオプションは始点アドレスを指定するため、OUTPUT チェーンには指定できない。

## コネクション・トラッキング拡張モジュール：-m state

コネクションの状態を指定するためのモジュールである。

--state コネクションの状態： コネクションの状態の名前を次の中から選ぶ。複数指定する場合は、, で区切って並べる。なお、大文字と小文字の区別はされない。

- NEW：新しく状態テーブルに追加されたパケット。
- ESTABLISHED：接続テーブルにエントリがあるパケット。
- RELATED：新しく状態テーブルに追加されたパケットであるが、既存の接続と関係があるもの（FTP のデータチャンネルへの接続や ICMP エラーなど）。
- INVALID：接続テーブルにエントリがないパケット。

-m state を指定すると、カーネルに ip\_conntrack モジュールが自動的に組み込まれる。このモジュールが組み込まれると、NF\_IP\_PRE\_ROUTING と NF\_IP\_LOCAL\_OUT のフックに関数が登録され、ここを通るコネクションの状態が追跡されるようになる。そして、state モジュールにより、コネクションの状態をフィルタリング・ルールに指定できるようになるため、結果として、ステートフル・インスペクション型のファイアウォールと同じくらいルールの記述が簡単になる。

コネクションの状態はカーネル内のテーブルで管理され、その内容は /proc/net/ip\_conntrack で参照できる。TCP コネクションの状態については、TCP ヘッダを調べることで、接続の開始から終了までの状態を順次みていくことができる。しかし、UDP や ICMP のように状態を保持しない通信の場合は、1 番目のパケットを接続の “開始” とみなせても、切断については明示的な手続きがない。そこで、ip\_conntrack では、テーブルの各エントリに設定したタイムアウト値をもとに切断を判断し、状態テーブルから削除する。

引数の コネクションの状態 に指定する名前は、TCP のコネクション状態とは無関係である。すなわち、SYN パケットが必ずしも NEW にマッチするわけではない。コネクションの状態 にはあくまでも状態テーブルを参照した結果を指定する。

### パケット拒否拡張モジュール：-j REJECT

マッチしたパケットに対して指定した ICMP <sup>18</sup> を返送するモジュールである。それ以外の動作は-j DROP と同様である。返送する ICMP のタイプは、--reject-with で指定することができ、オプションを省略した場合は Port Unreachable が返送される。

--reject-with ICMP タイプ：返送する ICMP タイプの名前を次の中から選ぶ。

- icmp-net-unreachable
- icmp-host-unreachable
- icmp-port-unreachable
- icmp-proto-unreachable
- icmp-net-prohibited
- icmp-host-prohibited

TCP の場合に限り、上記のタイプに加えて、RST パケットを返信する以下のタイプを指定することができる。

- tcp-reset

### ロギング拡張モジュール：-j LOG

パケットのログを syslog に記録するためのモジュールである。iptables ではターゲットとして実装されている。

--log-level ログのレベル：syslog の優先順位 (priority) を指定する。指定できる優先順位については、syslog.conf のマニュアルを参照されたい。なお、デフォルトのレベルは warning である。

--log-prefix プレフィックス文字列：このオプションを指定すると、記録されるログの先頭にプレフィックス文字列が追記される。プレフィックス文字列に指定できる文字列の最大長は 29 文字である。攻撃名などが記録されるようにしておくと、grep で検索するときに便利である。

--log-tcp-sequence：TCP のシーケンス番号を記録に取ることを指定する。

--log-tcp-options：TCP のオプションを記録に取ることを指定する。この場合、オプションの内容は 16 進表記 (HEX) で記録される。

### 3.5.2 フィルタの表示

iptables コマンドでフィルタのリストを表示する書式を以下に示す。

```
iptables -L [チェーン名] [-v] [-x] [-n] [--line-numbers]
```

-L オプションだけ指定すると、ユーザ定義チェーンを含むすべてのチェーンのルールが表示される。一方、-L のオプションの引数にチェーン名を指定すると、そのチェーンのルールだけが表示される。ただし、ここで表示されるのはルールの一部である。すべての情報を表示するには、-L オプションに加えて-v オプションも指定する。このリストでは、フィルタリング・ルールの内容だけでなく、そのルールにマッチしたパケットの数と総バイト数のカウンタも表示される。特に、ビルトイン・チェーンについては、チェーンに入ってきたパケットとバイト数も表示される。

-v オプションで表示されるバイト数は、数字の桁数が大きくなってくると自動的に K (1,000)、M (1,000K)、G (1,000M) のように丸められていく。これでは、統計をとる場合などのように正確な数字を知りたいときには不便である。そこで、さらに-x オプションも同時に指定することで、正確な数字の表示が可能になる。

<sup>18</sup>ICMP (Internet Control Message Protocol)：IP プロトコルを補助するためのプロトコルであり、通信障害があった場合の送信元ホストへの障害報告や、ネットワークの通信確認の診断に使用される。通信確認のために頻繁に使用される ping コマンドは、ICMP を利用している。

-I オプションや-D オプションでルール番号を指定してルールを挿入・削除するときには、先頭からルールを数えることは面倒である。そこで、`--line-numbers` オプションを指定することで、各行の先頭にルール番号が表示される。

-L オプションを付けると、デフォルトでは自動的に IP アドレスからホスト名、ポート番号からサービス名へと変換した上で表示される。場合によっては IP アドレスやポート番号を直接見たい場合もある。そのような場合には、`-n` を指定することで、変換せずにそのまま IP アドレスとポート番号を表示することができる。

### 3.5.3 ポリシーの設定

ポリシーとは、どのフィルタリング・ルールにもマッチしなかった場合、デフォルト動作として働くターゲットのことである。ただし、ポリシーを設定できるのはビルトイン・チェーンだけであり、ユーザ定義チェーンには設定できない。ポリシーに設定できるターゲットは `ACCEPT`、または `DROP` のどちらかだけであり、他のチェーンをポリシーに設定することはできない。チェーンにポリシーを設定する書式を次に示す。

```
iptables -P { INPUT | OUTPUT | FORWARD } { ACCEPT | DROP }
```

ユーザ定義チェーンのデフォルトの動作は、そのユーザ定義チェーンを呼び出したチェーンに戻ることである。ユーザ定義チェーンにポリシーを設定したい場合には、すべてにマッチするルールを最後尾に追加すればよい。

### 3.5.4 チェーンの作成と削除

iptables コマンドでチェーンを作成・削除する書式を次に示す。

(a) 作成 `iptables -N チェーン名`

(b) 削除 `iptables -X チェーン名`

チェーンを作成する場合のチェーン名は 29 文字以下の文字列であり、すでにあるチェーン名と重複しなければなんでもよい。ただし、無難な文字（英数字）にした方がよい。なお、大文字と小文字は区別される。英数字以外の名前を持つチェーンを作成しても、パケット・フィルタリングは問題なく機能しているように見えるが、どこにどのような不具合が潜んでいるか分からないので、英数字以外の文字を使用することは避けるべきである。また、チェーンがループしてしまう場合は、iptables がそれを検出して登録（-I）を拒否するようである。

作成したユーザ定義チェーンを削除する場合には、以下のことが必要である。

- ルールが登録されていない。
- 他のルールのターゲットに指定されていない。

## 3.6 アドレス変換の設定

アドレス変換ルールを追加・編集するには、iptables コマンドに `-t nat` オプションを指定して実行する。

### 3.6.1 IP マスカレードルールの編集

IP マスカレードのルールを登録・編集をする書式を次に示す。

(a) 作成

```
iptables -A POSTROUTING -t nat [-s 始点 IP アドレス] [-o インタフェース名]  
-j MASQUERADE
```

(b) 削除 `iptables -D POSTROUTING ルール番号 -t nat`

(c) 表示 `iptables -L POSTROUTING -t nat`

インターネットに出て行くすべてのパケットを IP マスカレードの対象とするには次のように記述する。

```
iptables -A POSTROUTING -t nat -o 外側向けインタフェース名 -j MASQUERADE
```

フィルタリング・ルールの設定の際に説明したオプションを駆使することで、IP マスカレードの対象と“する”・“しない”パケットの種類を細かく指定できる。しかし、現実にはそのようなルールを書くことはほとんどないので、上記の書式は簡略化してある。

### 3.6.2 Destination NAT ルールの編集

パケットがインタフェースから出て行く直前に、パケットの終点アドレスを書き換える NAT ルールの書式を次に示す。

(a) 作成

```
iptables -A PREROUTING -t nat [-p プロトコル] [--dport 終点ポート番号]
[-i インタフェース名] [-s 始点 IP アドレス] [-d 終点 IP アドレス]
-j DNAT
--to-destination IP アドレス[-IP アドレス] [:ポート-ポート]
```

(b) 削除 `iptables -D PREROUTING ルール番号 -t nat`

(c) 表示 `iptables -L PREROUTING -t nat`

netfilter では、このタイプの NAT のことを **Destination NAT** (DNAT) [11] と呼び、プライベート・ネットワーク上のサーバをインターネットに公開するときに使用する。例えば、グローバル・アドレスが 1 つしかないネットワークでサーバを外部に公開するには、外部からのアクセスを内部ホストへ中継する必要がある。この場合、特定のポートへ送られてきたパケットを内部ネットワークに転送するときに、パケットの終点アドレスをサーバホストのアドレスに書き換える（同時にポート番号を書き換える場合もある）。内部の Web サーバ（アドレス `xx.xx.xx.xx`）を公開するには、次のように設定する。

```
# iptables -A PREROUTING -t nat -p tcp --dport 80 -i 外側向けインタフェース名 -j DNAT \
--to-destination xx.xx.xx.xx
```

## 3.7 ログ管理

サイトの外部と内部との間の通信が集中するファイアウォールは、通信の記録（ログ）を収集するのに適したポイントでもある。攻撃の痕跡を記録したログは、セキュリティ・インシデントに対応するときに極めて重要な意味を持つ。

ファイアウォールでのログ収集の目的は、主に次の 2 つである。

1. 通信量やユーザが利用するサービスの傾向を分析する。
2. 攻撃元や攻撃方法を調べる。

ファイアウォールを経由する通信の傾向を把握するのは、ファイアウォール管理者の重要な仕事の 1 つである。例えば、時間帯や曜日ごとのアクセス数、転送バイト量、どのサービス（プロトコルの種類やポート番号）がよく利用されているか、ユーザ認証を行っている場合はその頻度や利用ユーザ名などの情報をこまめに調べ、“定常状態”を把握しておく。そして、いつもと異なる傾向（ログの量の増加など）に気付いたら、必ず原因を調査する。なぜなら、何らかの攻撃を受けているかもしれないからである。

このように、ファイアウォールが記録するログから通信の傾向を把握できるが、そのために毎日“生の”ログを読むのは辛い作業である。現在市販されている多くのファイアウォール製品には、ファイアウォールのログを解析し、見やすく整形して出力するレポーティング機能が含まれている。あるいは、WebTrends Firewall Suite などのレポーティング専用ツールもある。よって、一般的にはこれらの製品を利用するとよい。

さらに、ファイアウォールがアクセスを拒否したときの情報をログに残すように設定しておくことで、攻撃の頻度や攻撃元、攻撃方法などを分析するときに有用なデータとなる。

以上のように、通信の傾向を管理者が把握したり、攻撃を受けたときに解析を行うためにも、必ずログを記録するように設定しておくべきである。

ちなみに、付録 [A.1](#) の中で NAT のログをとるようになるには、以下を追記すればよい。これにより、内部から外部への不正アクセスが発生した場合に接続機器の特定が可能となる。

```
# iptables -I FORWARD -m state --state NEW -j LOG
```



## 4 Linux システム概説

Linux は、UNIX 系の OS の一つであり、組み込み用からサーバ用途まで広い範囲で利用されている。この章では、Linux 誕生に至るまでの経緯を概観し [14],[16]、その後、実験で実際に利用する CentOS 7 を例として、Linux の動作、及び管理に関して説明する。実験の目的上、システム管理者にとって必要となる知識を中心に解説している。尚、Linux 管理に関する記述の一部は文献 [17] からの引用である。

Linux コマンド構文等の記載においては、任意のオプションを意味する `[]`、排他的選択を意味する `{}`、`|` 等の記法を用いている。

### 4.1 Linux の誕生

#### 4.1.1 UNIX の誕生と普及

1960 年代後半において、IBM System/360 シリーズの汎用大型計算機（メインフレーム）は、市場を独占するほどの成功を収め、商用・技術計算用を問わず広く利用されていた。メインフレームでは、通常 TSS (Time Sharing System) により、複数のユーザが一台の計算機を共同利用する形態を取っていた。一方、MIT (Massachusetts Institute of Technology)、GE 社 (General Electric Company)、BTL (Bell Telephone Laboratories、米国の通信電話会社 AT&T 傘下の研究所) は、次世代のメインフレーム OS を目指した Multics の共同開発を行っていた。1960 年代末において、Multics の開発は進まず<sup>19</sup>、1969 年に BTL は Multics の開発から撤退した。Multics は当時のハードウェアで実現するには巨大過ぎたと言われている。

この頃、集積回路技術の進展により、メインフレームより遙かにコンパクトで低価格なミニコンピュータと呼ばれる計算機が登場しつつあった。DEC (Digital Equipment Corporation) の PDP/VAX シリーズはその筆頭である。1967 年修士課程修了後 BTL に入社し、Multics の開発に従事していた Ken Thompson は DEC の PDP-7 に Space Travel ゲーム（天体シミュレータ）を移植するために、Multics の思想を一部取り入れた OS である UNIX を開発した。その後、社内プロジェクトとして PDP-11 を入手し、Dennis Ritchie 等の協力で 1971 年にミニコンピュータ PDP-11 上に UNIX First Edition が完成した。UNIX は当初アセンブラで記述されていたが、1971 年以後システム記述を目的として C 言語が開発され、カーネルが書き改められた。ここに、現在でも広く利用されている高級言語の一つである C 言語と、高級言語で記述された初期の OS の一つである UNIX が誕生することとなった。

UNIX は当初 BTL の社内プロジェクトとして開発されたこともあり、商品として積極的に販売されていたわけではない。UNIX はソースプログラムも低価格でライセンスされていたため、大学等で独自に開発が進み。その中でも UCB (University of California, Berkeley) では、vi エディタの作者として有名な Bill Joy 等による UNIX の改良・機能追加版である BSD (Berkeley Software Distribution) が精力的に開発・配付されていた。BSD 版 UNIX は、TCP/IP 等の重要な機能を実現しており、多くのオープンソース UNIX (FreeBSD 等) の元になった。一方で、UNIX の商標所有者であった AT&T は 1988 年に UNIX System V Release 4 (SVR4) を発表し、現在まで続く多くの商用 UNIX (Solaris, HP-UX 等) の元となった。このため、1990 年代以降、UNIX の大きな系統として、BSD 系と System V 系の二つが存在する。国際規格 POSIX は多くの UNIX バージョン間の相違を解消しようとする試みであるが、現在に至るもその影響が残っている。

X Window System 開発チームの一員が著した “The UNIX Philosophy” [15] には、UNIX 哲学の第一から第三として、

1. 小さいものは美しい (Small is beautiful.)
2. 一つのプログラムには一つのことをうまくやらせる (Make each program do one thing well.)
3. できるだけ早く試作する (Build a prototype as soon as possible.)

が挙げられている。UNIX はその初期の頃から既に、小さなユーティリティプログラムの組み合わせとテキスト入出力のパイプ処理、シェルスクリプト等の仕組みによって、様々な処理を実現することが基本哲学であり、これらは Linux に受継がれている。UNIX/Linux システムの管理においても、様々な設定は設定テキストファイル

<sup>19</sup>一説では 1000 人のユーザを目的としていたのに、現実には 3 人ですら満足に利用できなかったとか [14]

と制御用コマンドで制御し、実行結果のログファイルもテキスト出力され、テキスト処理を介した様々なアプリケーション連携や自動化を実現可能とすることが一般的である。各自のシステムに特有な管理作業は、ユーティリティコマンドとスクリプトの組み合わせ（初期には sh+sed/awk, その後 Perl, 近年では Python）によって実現し、cron による自動実行後、mail による結果報告を得る、という手順が一例となる。しかし注意して欲しいことは、この恩恵を受けるためには、CLI (Command Line Interface) とエディタによるテキスト編集処理が必須であり、今回の実験でも CLI を基本とする。尚、簡単な UNIX/Linux コマンドの説明を附録に示すが、詳細は man コマンドを各自参照することとする。

#### 4.1.2 GNU (GNU's Not Unix)

UNIX 初期の開発者の多くは Ken Thompson や Bill Joy のように、業務上というよりは自分自身のために UNIX 開発作業に携わっていたことに注意して欲しい。しかし、UNIX の著作権は AT&T にある。AT&T は 1984 年の反トラスト法訴訟での和解・分割以降は、既に広く普及していた UNIX 販売を積極的に推し進め、UNIX の自由な改変・配付はしだいに困難となっていく。これに対して Richard Stallman は“フリーソフトウェア”による OS 実現を目指して、1983 年 GNU (GNU's not Unix) プロジェクトを開始した。GNU のソフトウェアは UNIX コマンドの正確な再実装というよりは、むしろ上位互換の物も多く、商用 UNIX ユーザも GNU プロジェクトの成果の一部を利用することが一般的であった。bash や gzip もそうであるが、特に、コンパイラ gcc とエディタ emacs は、その高性能さが認められている。また、GNU プロジェクトのもう一つの著名な成果物は GPL (GNU General Public License) であり、現在でもオープンソースプロジェクト成果物のライセンスに使用されている。GNU プロジェクトはフリーな OS 実現が目標であるため、OS カーネルである GNU Hurd の開発も行っているが、Hurd は今現在も開発版の段階にある。

#### 4.1.3 Window System

コンピュータハードウェア価格の低下とともに、高解像度のビットマップディスプレイとイーサネットによる高速なネットワークング、(一部では RISC ベースの) 比較的高速な CPU を備え、OS には UNIX を搭載した“ワークステーション”と呼ばれる計算機が登場した。筆頭の Sun Microsystems 社、コンピュータグラフィックス分野で有名となった Silicon Graphics 社 (SGI) 等、多くのメーカーが現れ、また自社技術のオープン化、共通仕様化によって、今日まで残る影響を与えた企業も多い。これらのワークステーションでは、高解像度ディスプレイを利用した GUI (Graphical User Interface) を実現するために、通常 X Window System が利用された。X Window System (X, X11) は 1984 年 MIT で開発が開始された GUI 環境構築のためのクライアント・サーバ方式のシステムである。その柔軟な設計思想は、様々な環境下での利用を可能としている。X11 はユーザインタフェースを規定していないため、利用者のルック&フィールを統一するために、共通 GUI ライブラリに基づくデスクトップ環境 (GNOME, KDE 等) と呼ばれるアプリケーション環境を介して X11 を利用することが一般的である。X11 は最近まで UNIX/Linux ではほぼ標準的に利用されてきたが<sup>20</sup>、近年ようやく代替システムとなる Wayland が開発され、利用可能な状況にある。一般利用者には必要不可欠な GUI 環境ではあるが、4.3.4 節でも述べるように、UNIX/Linux システムにおいては OS の一部ではなくオプション扱いである。すなわち、GUI 環境無しの動作モードが選択可能である事を強調しておく。

#### 4.1.4 IBM PC-AT 互換機の普及と Linux の誕生

パーソナルコンピュータの分野では、1984 年 IBM 社が発売した PC/AT 機は、アーキテクチャのオープン化戦略により、多くの PC/AT 互換機が価格競争する市場となった。PC/AT 互換機と Microsoft 社の OS MS-DOS の組み合わせは、パーソナルコンピュータ分野のデファクトスタンダードを占めるようになったが、もはや IBM のリーダーシップは失われており、新アーキテクチャ PS/2 の提案は失敗に終わった。その後の PC 市場は、CPU メーカー Intel と OS メーカー Microsoft の主導で進展することとなる。1995 年に発売された Windows 95 は、初めて TCP/IP スタックを標準装備する Windows であり、一般ユーザにおけるインターネット時代幕開けの象徴で

<sup>20</sup>当然例外は存在する。最大の例外は NeXT の Display PostScript の流れをくむ Mac OS X (現在の macOS) の Quartz であろう。

もある。PC は当時最も低価格・低性能なレンジの計算機ではあったが、最も台数の多いレンジでもある。その後の x86 アーキテクチャ CPU 性能の順調な向上は、現在ではワークステーションレンジの計算機を一掃し、これに取って代わることとなった。

1980 年代後半から 1990 年代にかけては、ワークステーション全盛の時代であり、大学や企業では UNIX ワークステーションが利用されていた。一方で、各家庭では遙かに低性能な PC 上で主に MS-DOS や Windows が利用されていた。UNIX も次第に PC (x86 アーキテクチャ) へと移植されていった。AT&T からライセンスされた商用 UNIX (XENIX, Solaris 等)、BSD 系のオープンソース UNIX (FreeBSD, NetBSD 等)、さらにアムステルダム自由大学の Tanenbaum 教授による独自の UNIX 系の教育用 OS MINIX も利用可能であった。そして、1991 年ヘルシンキ大学の学生 Linus Torvalds の開発による UNIX 系 OS カーネルがインターネットの MINIX ニュースグループ (comp.os.minix) に投稿され、ここに Linux が誕生した。Linux は当初最低限の機能しか持たない小さな OS カーネルであった。C 言語とアセンブラで 1 万行程度のプログラムであったようである。しかし、ちょうどこの時期 BSD 系の UNIX は AT&T との訴訟に関連して、権利関係に問題を抱えていた。MINIX は教育用 OS であり、利用も制限されていた。1992~1994 年にかけては、GPL で自由に利用可能な UNIX 系 OS は Linux のみであり、Linux が X Window System やネットワーク機能を備え、実用的とされるバージョン 1.0 を 1994 年にリリースする頃には多くのユーザが集まっていた。

#### 4.1.5 Linux ディストリビューション

“Linux” という用語はしばしば各種ソフトウェアを含んだ OS システム全体を指して利用される場合も多いが、厳密には Linux は OS のカーネル部分のみであり、Linux カーネルに組み合わせるソフトウェアによって、各システムの挙動は異なるものとなる。OS のカーネル以外の部分に関しては、GNU プロジェクトのプログラム群が利用可能であり、GUI 環境には X11 Window System が利用可能である。しかし、さらに上位のアプリケーションプログラム、例えばワープロや表計算ソフト等、あるいは WWW サーバや開発環境等に関して、様々な選択肢がありうる。しかも、そもそも独立に開発され、ソースプログラムで提供される Linux カーネル、GNU/X11 ソフトウェア、各種アプリケーションプログラム等の依存関係を解決し、動作可能な実行ファイルにビルドするだけでも一苦勞である。さらに、これらの実行可能ファイルを起動可能な状態に PC に配置・環境設定することは、一般ユーザにはあまりにも敷居が高すぎる。Linux が一般ユーザに普及していくためには、この様な作業を代わって準備してくれる Linux ベンダーやコミュニティの存在が必要不可欠であった。このように容易にインストール可能な状態でパッケージ化された Linux システムを Linux ディストリビューションと呼ぶ。

初期の Linux システムの 3 大ディストリビューションは、

1. Slackware
2. Red Hat
3. Debian

であった。Debian はコミュニティ（無償のボランティア）ベースのディストリビューションであり、現在でも開発が続いているが、むしろ Debian をベースとした Ubuntu の方が有名となっている。Red Hat 社は当初ディストリビューションのバイナリコードを無償配付しつつ、サポートを有償提供するビジネスモデルを取る Linux ベンダーであった。しかし今では、企業向けにサポート期間を長期化した Red Hat Enterprise Linux (RHEL) を有償販売している。また、RHEL の GPL 公開されたソースコードを元に生成された RHEL クローンディストリビューションが存在し、本実験で利用する CentOS はその一つである。

## 4.2 デバイスとファイルシステム

この節では、Linux が各種のハードウェア装置を統一的に扱う“デバイス”という概念を説明し、特に重要なストレージデバイスの扱いに関して述べる。

### 4.2.1 デバイスファイル

UNIX/Linux では、多くの装置へのインターフェイスはデバイスファイル（デバイスノード）として扱い、通常のファイル入出力と同様に扱うことを可能としている。Linux では、udev デーモンがデバイス名を決定し、/dev ディレクトリ以下に動的にデバイスノードを作成する。デバイスノードには、以下で説明する HDD, DVD-ROM 等のストレージデバイス以外に、4.4.1 節で説明するネットワークデバイス、プリンタ、疑似乱数生成器等も含まれている。

### 4.2.2 ファイルシステム

Linux システムは通常 HDD/SSD などのディスクに保存されており、インストール時に構成が決定されるため、ファイルシステム作成の詳細に関与する必要性はやや低いため、以下では簡単に触れることとする。

1 台のディスクを複数の領域に分割し、独立して操作できるようにした区画をパーティションと呼ぶ。現在 MBR と GPT の二種類のパーティション形式が存在する。MBR では最大 2TiB までの容量しか対応できないため、今後は GPT が主流と成っていくものと思われる。MBR 形式のパーティショニングには fdisk を利用する。一方 GPT に関しては開発途上であり、最も一般的なツールである GNU Parted を利用すれば、パーティションテーブルの確認、パーティションの変更等が可能である。どのようにパーティション分割を行えば良いかに関しては、ルートファイルシステムのためのパーティション以外に、以下の 2 つのパーティションが必要であることを除けば、特に制約はなく、各システムの要求に応じて構成すればよい<sup>21</sup>。

1. swap 領域に利用するパーティション
2. /boot ファイルシステムに利用するパーティション

作成されたパーティションには、udev デーモンによって固有のデバイスノードが割り当てられる。SATA ディスクは、/dev/sda, /dev/sdb, ... が、ディスクパーティションは /dev/sda1, /dev/sda2, ... がデバイスノードとなる。ディスクパーティションはそのまま利用することはできず、ファイルシステムを生成する必要がある。Linux 標準のファイルシステム形式は ext3, ext4 であるが、それ以外にも xfs, btrfs などの様々なファイルシステム形式が利用可能であり、それぞれ異なる特徴を有している。

UNIX/Linux では、すべてのファイルシステムは、ルートファイルシステムからの階層ファイルツリーとして、木構造のどこかに位置する。従って、ルートファイルシステム以外のファイルシステムは、/etc/fstab に記述されたマウントポイントにマウントされて、全体の部分木を構成することになる。このように、各パーティションのデバイスノードを指定し、適当なファイルシステムを生成後、ルートファイルシステム下のマウントポイントにマウントすれば、OS から自由に利用可能となる。デバイスノードとマウントポイントの対応は /etc/fstab に記載するが、最近では一意の UUID がデバイスノード名の代わりに利用されている。この場合、blkid コマンドを用いて各デバイスノードの UUID を確認しなければ、マウントポイントとデバイスとの対応関係を知ることができない。

ディスクパーティション上に直接ファイルシステムを生成して利用している場合には、その後の構成変更に関してあまり自由度がない。このため、CentOS 7 (RHEL 7) では標準的に LVM (Logical Volume Manager) を利用して、ファイルシステムを作成している。LVM では複数のディスクパーティションから伸縮自由な論理ボリューム LV (Logical Volume) を構成し、LV 上にファイルシステムを作成する。lvdisplay コマンドによって、LV の詳細情報を表示することが可能である。尚、CentOS 7 (RHEL 7) では、LV を直接マウントせず、デバイス Mapper を経由してマウントしているため、/etc/fstab の内容は、/etc/mapper/〇〇 となっている。デバイス Mapper で変換されたデバイスと LV との関係は、デバイス Mapper コマンドの dmsetup ls による表示内容と lvdisplay コマンド実行時の Block device 表示を対応させることで確認が可能である。

### 4.2.3 ディレクトリ構成

CentOS を含む多くの Linux ディストリビューションでは、Linux Foundation が管理する FHS (Filesystem Hierarchy Standard) に従ったディレクトリ構造を利用している (表 5)。ディレクトリ構成の説明は、オンライン

<sup>21</sup>環境によっては、これら二つのパーティションも必要ない場合がありうる。



表 5: ディレクトリと役割 [17]

ディレクトリ	役割
/boot	システム起動時に必要なブートローダ関連のファイルやカーネルイメージを配置
/dev	システム起動時に接続されているデバイスから自動作成されるデバイスファイルを配置
/etc	システム管理用の設定ファイルや、各種ソフトウェアの設定ファイルを配置
/var	システム運用中にサイズが変化するファイルを配置
/tmp	アプリケーションやユーザが利用する一時ファイルを配置
/usr	ユーティリティ、ライブラリ、コマンド等のユーザが共有するデータを配置
/bin	一般ユーザ、管理者が使用するコマンドを配置
/sbin	主に管理者が使用するコマンドを配置
/lib, /lib64	/bin や /sbin 等におかれたコマンドやプログラムが利用するライブラリを配置
/home	ユーザのホームディレクトリを配置
/root	root ユーザのホームディレクトリ
/proc	カーネルやプロセスが保持する情報を配置

ンマニュアル `man hier` でも確認することが可能である。尚、CentOS では、システム設定ファイルを集めたディレクトリ `/etc/sysconfig` が存在し、一部の重要なシステム設定はこのディレクトリ内に格納されている。ディレクトリ構造は、静的（システム管理者の操作以外では変更されないバイナリファイル、ライブラリ、ドキュメント等）であるか可変（システム稼働中に変更されるファイル）であるかも考慮して分類されている。システム管理者が特に注意すべきディレクトリは、静的な設定ファイルが含まれる `/etc` と様々なプロセスの出力ファイルが含まれる `/var` である。

このように、Linux のファイル／ディレクトリはすべて `/` を起点としたツリー上に存在するため、`/` で始まる絶対パスにより一意に指定可能である。一方で、コマンドを実行する際には、実行時の起点となるワーキングディレクトリ (Current Working Directory) が必ず存在し、CWD からの相対的な位置としてファイル／ディレクトリを指定することも可能である。これを相対パスと呼ぶ。各ユーザの作成したファイル／ディレクトリは、該当ユーザの UID/GID (4.7 節参照) を属性として所有し、所有ユーザ／同一グループ／それ以外のユーザの 3 種類の対象毎にファイルへのアクセス権（読み取り／書き込み／実行）が設定される。アクセス権（パーミッション）は、`umask` コマンド、`chmod` コマンドでコントロールすることができる。

Linux には、ファイル／ディレクトリ以外にリンクが存在する。リンクの中でも、シンボリックリンクは多用されているため、システム管理者は念頭に置いておく必要がある。シンボリックリンクは、大部分のコマンドに対して、単なるファイル／ディレクトリのように振る舞うが、実際には単に実在のファイル／ディレクトリの絶対／相対パスを示しているに過ぎない。シンボリックリンクを削除しても、リンク先のファイル／ディレクトリは削除されることはない。対象の種類（ファイル／ディレクトリ／リンク）とパーミッションは、`ls` コマンドに `-l` オプションを付けて実行することで確認することができる。

Linux ディストリビューションのインストーラは、ストレージデバイス上に上記のファイルシステム、及びブートローダを配置し、Linux システムを起動可能な状態とする。次節では、インストールが完了した後、電源投入後の動作を順番に見ていくこととする。

### 4.3 Linux の起動プロセス

この節では、Linux システムが起動するまでのプロセスを順を追って見ていく。先述したように UNIX/Linux システムでは様々な作業を実現する場合に複数の選択肢がありうる。これは異なったディストリビューション間でも違いうるが、同一ディストリビューション内でもバージョン毎に相違する可能性がある。従って、以下の記述においては、CentOS 7 (RHEL 7) を例とした説明であることに注意して欲しい。

#### 4.3.1 ブートローダ

PC の電源を投入すると、まず BIOS/UEFI が実行され、設定されている起動媒体（通常 HDD）のブートローダが実行される。ブートローダは OS カーネルを読み込むための小さなプログラムであり、Linux システムでは GRUB2 が多く利用されている。GRUB2 はルートファイルシステム等のパラメータと共にブートメニューで指定された Linux カーネルを起動する（図 17）。注意すべき点は、GRUB2 が起動する Linux カーネルイメージ（`vmlinuz`-バージョン番号等）と後述する初期 RAM ディスクイメージ（`initramfs`-バージョン番号等）を GRUB2 から読み込み可能な場所に配置すべきことである。このため、これらのファイルの格納場所にはソフトウェア RAID（冗長性を持ったディスクシステム）や LVM（論理ボリューム）などの高機能化されたデバイスは利用できず<sup>22</sup>、独立した `/boot` パーティションに保存される。GRUB2 設定を間違えると Linux システムが起動不能となるが、通常はインストール時、カーネル更新時共に自動処理されるため、システム管理者の誤った介入がなければ考慮する必要は無い。

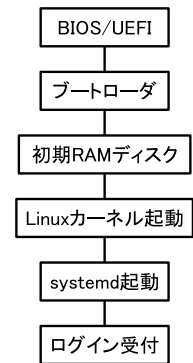


図 17: 起動プロセス

#### 4.3.2 カーネルの起動

ブートローダによって、Linux カーネルと初期 RAM ディスクがメモリに配置されカーネルが起動されると、カーネルはルートファイルシステム（ファイルシステムツリーのルート `/`）をマウントしようとする。しかし、ここで問題が発生する。Linux カーネルは、今や非常に広範なデバイスをサポートしており、ソフトウェア RAID、LVM、NFS (Network File System)、暗号化ファイルシステム等、様々なデバイスがルートファイルシステムになりうる。これらすべてを単一カーネルファイルで対応することは不経済であるため、それらのハードウェアドライバ等はカーネルに含めずルートファイルシステム内に格納される。しかし、カーネルにドライバが含まれないルートファイルシステムを利用している場合には、カーネル起動時のみはルートファイルシステムが利用不可能となる。このため、ルートファイルシステムマウントに必要なプログラム類を初期 RAM ディスクに格納し、カーネル起動時のみこれを利用する。初期 RAM ディスクのプログラムでルートファイルシステムをマウント後はルートをそちらに切り替えて、初期 RAM ディスクは不要となり、Linux カーネルが正しい構成で起動されたこととなる。

#### 4.3.3 最初のユーザプロセス `init`/`systemd`

カーネル起動後、最初（PID 番号 1）のプロセスが起動される。最初のプロセスは、UNIX の時代から `init` と呼ばれるプログラムが担当しており、他のデーモンの起動処理を行う。デーモン (daemon) とは、常時バックグラウンドで起動しており、OS の機能を実現するプロセス（表 6 のサービス等）のことである。`init` はシステム起動時に必ず実行され、その処理時間が起動時間に大きく影響する。CentOS では `SYSV init`、すなわち UNIX System V 系の `init` が利用されてきたが、起動時間短縮の目的もあり、CentOS 7 より `systemd` に変更された<sup>23</sup>。

#### 4.3.4 `systemd` の機能／サービス管理

`init` はプロセス管理を行うデーモンであったが、`systemd` はそれよりも遙かに多くの機能を有するソフトウェア群と言える。実際、この点が 4.1.1 節で述べた UNIX 哲学と相容れないという批判が起こり、激しい議論を巻き起こした。とは言え、現在はほとんどの主要ディストリビューションで `systemd` が採用されている。以下では、`systemd` でサービスと呼ばれているデーモンプロセスの管理機能に関して主に説明することとする。

##### (1) ランレベル

`SYSV init` では、システムのランレベル（表 7）を定義し、各ランレベル毎に起動するデーモンを変更していた。通常は利用目的に応じて、サーバ用途では GUI が不要なためランレベル 3 で起動し、デスクトップ環境を利用す

<sup>22</sup>正確には、RAID1 のみは対応している。

<sup>23</sup>CentOS 6 では `SYSV init` と互換性の高い `Upstart` が一時的に使われていた。

表 6: systemd で管理されるサービスの例

systemd サービス	説明
chronyd.service	NTP (Network Time Protocol) サーバ/クライアント
crond.service	コマンドスケジューラ
dbus.service	D-Bus システムメッセージバス
NetworkManager.service	ネットワーク設定を行うデーモン
postfix.service	MTA (Mail Transfer Agent), メールの送配信デーモン
sshd.service	SSH サーバ
dhcpcd.service	DHCPv4 サーバ
dovecot.service	IMAP/POP メールサーバ
named.service	DNS サーバ
httpd.service	Apache HTTP サーバ

表 7: systemd のターゲットと SYSV init のランレベル [17]

systemd ターゲット	init ランレベル	内容
default.target	デフォルトランレベル	デフォルトターゲット
poweroff.target	ランレベル 0	電源オフ
rescue.target	ランレベル 1	レスキューモード
multi-user.target	ランレベル 3	マルチユーザモード
graphical.target	ランレベル 5	グラフィカルモード (マルチユーザ+GUI)
reboot.target	ランレベル 6	システム再起動

る目的であればランレベル 5 を、保守時には他ユーザの介入を避けるためにランレベル 1 のシングルユーザモードで起動するものである。systemd でも、ほぼこの定義を踏襲しているが、ランレベルとは呼ばずターゲットと称している。システム起動時のデフォルトターゲットは、

- (a) 確認 `systemctl get-default`

(b) 設定 `systemctl set-default multi-user.target`

のように、確認/設定が可能である。

また、systemctl のサブコマンド `isolate` を利用することで、`systemctl isolate target` のように、現在のターゲットを *target* に変更することができる。尚、従来のランレベル変更コマンドである `shutdown` (`shutdown -h now`), `reboot` コマンドもシンボリックリンクとして残されている。

## (2) サービス管理

systemd はシステム起動時のサービス起動以外に、システム動作時においても、systemctl コマンドを用いれば、様々なサービスの管理を行うことが可能である (表 8)。

## 4.4 ネットワーク管理

### 4.4.1 NetworkManager

NetworkManager は、有線/無線を問わずネットワークデバイスの管理を行うためのサービスである。GNOME control-center, `nmtool`, `nmcli` の 3 つの方法で設定を行うことができるが、以下では、CLI である `nmcli` (Net-



表 8: systemctl コマンドの利用方法 [17]

コマンド	サブコマンド [引数]	説明
systemctl	list-units	アクティブなすべてのユニットを表示
	list-unit-files	インストールされているすべてのユニットを表示
	status <i>service</i>	サービス <i>service</i> の状態を表示
	start <i>service</i>	サービス <i>service</i> を開始 (アクティブ化)
	stop <i>service</i>	サービス <i>service</i> を停止
	enable <i>service</i>	サービス <i>service</i> を enable (起動時自動開始) にする
	disable <i>service</i>	サービス <i>service</i> を disable (起動時停止) にする

表 9: nmcli コマンドの利用方法 [17]

コマンド	ターゲット	サブコマンド [引数]	説明
nmcli	device	status	ネットワークデバイスの状態表示
	device	show	ネットワークデバイスの詳細情報表示
	connection	show	接続情報の表示
	connection	show <i>connection</i>	指定した接続 <i>connection</i> の情報表示
	connection	add パラメータ	新しい接続の追加
	connection	delete <i>connection</i>	既存の接続 <i>connection</i> の削除
	connection	modify <i>connection</i> パラメータ	既存の接続 <i>connection</i> の編集
	connection	reload	すべての接続の再読み込み
	connection	up <i>connection</i>	指定した接続 <i>connection</i> の有効化
	connection	down <i>connection</i>	指定した接続 <i>connection</i> の無効化

workManager Command Line Interface) の利用方法について述べる。

nmcli コマンドの書式を以下に示す。

```
nmcli [オプション] ターゲット サブコマンド [引数]
```

よく利用されるターゲットとしては、デバイスの表示と管理を行う **device** (d と省略可) と接続の管理を行う **connection** (c と省略可) がある。物理的なネットワークインターフェイスに対応するデバイス名 (interface name, ifname) は、接続設定に付けられた接続名 (接続プロファイル名, ID) とは無関係であることに注意すること。ただし、通常デバイスと接続は 1 対 1 に対応しているため、接続名にはデバイス名と同一名称を付けることが多い。各ターゲット毎のサブコマンドを表 9 に示す。最後に **help** を付けることで利用方法／利用例を表示する。例えば、`nmcli c mod help` と入力する。

#### 4.4.2 ip コマンド

システム管理者に限らず、ネットワーク情報／ルーティング情報を確認するためには、**ip** コマンドを利用することができる。BSD 版 UNIX において、TCP/IP スタックが導入されて以来現在に至るまで、ネットワーク管理の目的で **ifconfig** コマンドが利用され続けている。しかし、Linux においては、残念ながら **ifconfig** の開発はストップしており、CentOS では利用が非推奨となっている。Linux では **iproute2** ソフトウェアの **ip** コマンドの利用が推奨されている。

**ip** コマンドの書式を以下に示す。**ip** コマンドはネットワーク管理／ルーティング管理に関する幅広い内容を含んでいるが、主なターゲットを表 10 に示す。

```
ip [オプション] ターゲット { サブコマンド | help }
```

表 10: ip コマンドの主なターゲット [17]

コマンド	ターゲット	説明
ip	address	IP アドレスとプロパティ情報の表示, 変更
	link	ネットワークインターフェースの状態を表示, 管理
	neighbour	arp テーブルの表示, 管理
	route	ルーティング情報の表示, 管理
	help	各ターゲットの help を表示

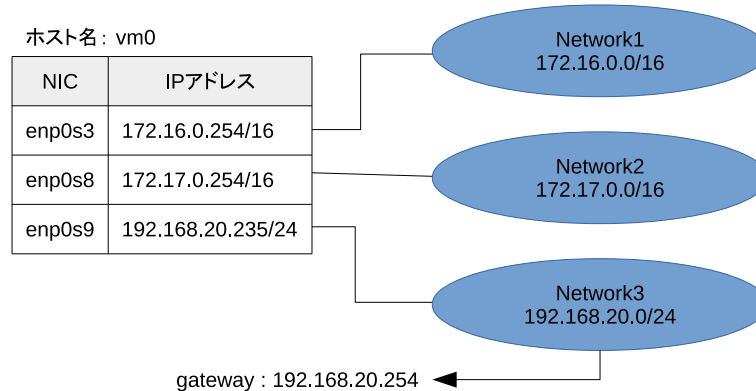


図 18: ネットワーク構成の例

`ip addr [show]` で各ネットワークデバイスの IP アドレスとプロパティ情報を, `ip route [show]` でルーティング情報を表示可能である. `ip` コマンドは前節で NetworkManager で述べたと同様にネットワークデバイスの設定やルーティング情報の設定にも利用可能であるが, `ip` コマンドによる設定内容の影響は永続的ではなく, 再起動後に失われる. 従って, 設定の変更に関しては, NetworkManager を利用すべきである.

ルーティング情報確認の例として, 図 18 のネットワーク構成において, 3 枚の NIC を有するホスト vm0 で, `ip route` コマンド, 及び, 従来利用されてきた `route` コマンドを実行した場合の実行結果を図 19 に示す. 図 18 のネットワークでは, Network3 に接続されているルータの IP 192.168.20.254 がデフォルトゲートウェイ (外部ネットワークへの経路) となっていることに注意する. `ip` コマンド, `route` コマンドいずれを利用して, vm0 が接続している 3 ネットワークへのルートとデフォルトルートの情報が表示されることが確認できる.

#### 4.4.3 firewalld

CentOS 7 では, 3.3 節で述べたパケットフィルタリング機能である Netfilter の設定ユーティリティとして, iptables 以外に firewalld が利用可能であり, 標準では firewalld を利用することになっている.

firewalld では, 典型的な利用環境を想定した設定が行われているゾーンと呼ばれるテンプレートが存在し, 適切なゾーンを選択して利用する, あるいは, 選択したゾーンに対してサービスの追加/削除をカスタマイズ設定して利用することで, 比較的容易に Netfilter の設定を行うことが可能である. 表 11 にゾーンを, 表 12 に firewalld の CLI 設定コマンド `firewall-cmd` の利用可能オプションを示す<sup>24</sup>.

`firewall-cmd` コマンドの書式を以下に示す.

firewall-cmd オプション
--------------------

このように, 用途の決まった比較的単純なルールのための利用環境においては, firewalld は簡単に設定が済み便利である. しかし, 複雑なルールを設定する場合には, リッチルール/ダイレクトルールと呼ばれる iptables と同様なルール設定を必要とし, 必ずしも iptables と比較して単純とは言えない.

<sup>24</sup>--permanent オプションを付けない場合, 設定内容はシステム再起動で失われる.

```
[root@vm0 ~]# ip r
default via 192.168.20.254 dev enp0s9 proto static metric 100
172.16.0.0/16 dev enp0s3 proto kernel scope link src 172.16.0.254 metric 100
172.17.0.0/16 dev enp0s8 proto kernel scope link src 172.17.0.254 metric 100
192.168.20.0/24 dev enp0s9 proto kernel scope link src 192.168.20.235 metric 100

[root@vm0 ~]# route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
0.0.0.0          192.168.20.254  0.0.0.0         UG      100    0      0 enp0s9
172.16.0.0       0.0.0.0         255.255.0.0     U       100    0      0 enp0s3
172.17.0.0       0.0.0.0         255.255.0.0     U       100    0      0 enp0s8
192.168.20.0     0.0.0.0         255.255.255.0   U       100    0      0 enp0s9
```

図 19: ip route コマンドの実行例

表 11: firewalld の定義済みゾーン

ゾーン	説明
block	外部からの接続はすべて拒否、内部から開始された外部への接続は双方向を許可
public	パブリックエリア用
external	外部ネットワーク用、マスカレードが有効
dmz	DMZ (DeMilitarized Zone) 用
internal	内部ネットワーク用
home	家庭用
trusted	すべての接続を許可

尚、複数の Network Interface が存在する場合に、各デバイスに対して、個別のゾーンを設定するためには、nmcli コマンドを利用し、connection.zone の値を設定することで実現可能となる。firewalld の --change-interface オプションは再起動後にはリセットされてしまうため、利用することができない。

## 4.5 CentOS のセキュリティ対策

かつて利用されていたネットワーク経由のサービスはほとんどが平文で通信内容をやり取りしていたが、セキュリティへの要求が高まるとともに、しだいに通信内容の暗号化が行われるようになってきた。この節では、代表的なプロトコルである SSH と SSL/TLS に関して述べる。これらのプロトコルでは、2.2 節で述べた暗号化方式を利用している。次に、システムのセキュリティ性能を大きく向上させる SELinux について概説する。

### 4.5.1 SSH (Secure Shell)

UNIX ワークステーション全盛の時代には、ネットワーク経由でリモート端末を利用するために、telnet や rlogin プロトコルを利用していた。しかし、これらのプロトコルは平文でユーザ名／パスワードを送信するため、盗聴の危険性が存在する。ssh (Secure Shell) プロトコル（ポート番号 22 を利用）では、サーバ・クライアント間の通信を共通鍵暗号による暗号化通信とすることで、盗聴の危険性を避けることができる。また、ユーザ認証方式もパスワード認証、公開鍵認証等いくつかの方式が提供されており、暗号化に関する知識を持たないユーザでもすぐに利用することが可能となっている。SSH プロトコルには、バージョン 1 と 2 が存在するが、バージョン 1 は脆弱性が知られており、バージョン 2 を利用することが好ましい。リモートシェルを提供する ssh 以外に、

表 12: firewall-cmd のオプション [17]

オプション	説明
<code>--get-default-zone</code>	デフォルトゾーンの表示
<code>--set-default-zone=zone</code>	デフォルトゾーンを <i>zone</i> に変更
<code>--zone=zone</code>	コマンド実行時のゾーンの指定
<code>--list-services</code>	ゾーンで許可されているサービスを表示
<code>--add-services=service</code>	ゾーンで許可するサービス <i>service</i> を追加
<code>--delete-services=service</code>	ゾーンで許可されているサービス <i>service</i> を禁止
<code>--permanent</code>	永続化を指定

ファイル転送を実現する `scp`, `sftp` も用意されている。尚, CentOS では, OpenBSD 開発チームによる SSH プロトコルのオープンソース実装 OpenSSH が利用されている。

リモートシェル `ssh` コマンドとリモートファイルコピー `scp` コマンドの書式を以下に示す。

- (a) リモートシェル `ssh` [ユーザ名@] ホスト名

(b) リモートファイルコピー `scp` ソースパス ターゲットパス  
(ただし, パスはローカルパスか [ユーザ名@] ホスト名: パス のいずれか)

#### 4.5.2 SSL (Secure Sockets Layer)/TLS (Transport Layer Security)[19],[20]

##### (1) SSL/TLS の概要

SSL はかつて WWW ブラウザのリーディングカンパニーであった Netscape Communications 社によって設計が開始された通信路暗号化プロトコルである。SSL は当初より WWW 以外の様々な TCP 上のプロトコルに適用可能となる事を念頭において設計されたため, 透過的 (transparent), つまり通信路上のデータを変更しないようになっている。Netscape Communications 社は, SSL v1.0, v2.0, v3.0 を開発し, その後, 標準化組織 IETF (Internet Engineering Task Force) において, SSL の後継として TLS v1.0, v1.1, v1.2, v1.3 が提案された。2018 年現在では, SSL 1.0 ~ SSL 3.0, TLS 1.0, TLS 1.1 にはすべて脆弱性が見つかっており, TLS 1.2 以降を利用することが望ましい。

SSL/TLS (以下 TLS と呼ぶ) は透過的なプロトコルであるため, 従来利用してきたプロトコルを TLS 上でそのまま利用した, HTTPS (HTTP over TLS) 以外の ○○○ over TLS プロトコルも次第に利用されるようになってきた。ただし, ○○○ over TLS プロトコルを利用する場合には, 通信の最初から TLS プロトコルに入るため, 平文で通信を行うポートとは別の番号のポートを用意する必要があると言うことを意味する。当然, この新しいポートにおいては平文のプロトコルは処理できないため, 両方のポートを維持する必要がある。これを避けるために, 最初は平文での通信を行い, 途中で STARTTLS コマンドにより TLS 暗号化通信を構成する STARTTLS 方式が従来プロトコルの拡張として存在する。STARTTLS を利用する場合には, 同一のポート番号で, 平文の通信も暗号化された通信も扱うことが可能となる。特にメール送信プロトコル SMTP においては, メールはいくつものメールサーバ間での中継後に送信先に届くことが多い。この場合, 最初の通信内容を暗号化しても, その後の経路での安全性を保証できないため, あまり意味が無い。このため, SMTP over TLS (smtps) はあまり普及せず, STARTTLS 方式による SMTP の暗号化が多く用いられている。

##### (2) サーバ証明書

通信の暗号化に利用される鍵が, 公開鍵, あるいは当事者間での交換に基づく鍵のみであった場合には, 中間者攻撃 (man in the middle attack) を完全に防ぐことは難しい。TLS では, サーバの身元を記載したサーバ証明書を, 第三者機関である“認証局 (CA, Certification Authority)”による署名により保証することで, 安全性を確保している。信頼できる CA の認証手続きは通常有料となり, この点が一般利用者への TLS 化普及の阻害要因の一つ

表 13: X.509 で利用される DN

要素	意味
C (Country)	国名
O (Organization)	組織名
OU (Organization Unit)	部門名
CN (Common Name)	一般名

## Certificate Request:

## Data:

Version: 0 (0x0)

Subject: C=JP, ST=Aichi, L=Nagoya, O=Nagoya University, OU=Computer Science,  
CN=example.ice.nuie.nagoya-u.ac.jp

## Subject Public Key Info:

Public Key Algorithm: rsaEncryption

Public-Key: (2048 bit)

Modulus:

00:.....

Exponent: 65537 (0x10001)

## Attributes:

a0:00

Signature Algorithm: sha256WithRSAEncryption

0f:.....

図 20: CSR の例

となっていたが、最近では Let's Encrypt 等の非営利無料証明機関も存在し、TLS 化の普及が促進されるものと思われる。サーバ証明書は、ITU-T (International Telecommunication Union Telecommunication Standardization Sector) の策定した X.509 に定められた公開鍵証明書の規格に従うものとする。

### (3) 公開鍵基盤 PKI (Public Key Infrastructure)

TLS に限らず、今後いっそうの増加が予想される電子取引においては、利用される公開鍵とその所有者との対応関係を保証することは重要である。この保証を与えるための基盤となるのが PKI である。PKI の中核をなすのは、証明書 (certificate) である。証明書とは、証明書所有者 (subject) の識別名 (DN: Distinguished Name, 表 13) に公開鍵を結びつけるものである。証明書は信頼できる第三者の証明機関 (CA: Certification Authority) が発行者 (issuer) となり、証明書所有者の証明書署名要求 CSR (Certificate Signing Request, 図 20) に対して発行される。証明書には、所有者の識別名と公開鍵、発行者の識別名、証明書の有効期間等の情報が含まれている (図 21 参照)。作成された証明書は、ハッシュ関数によりメッセージダイジェスト (フィンガープリント) が計算され、発行者の秘密鍵によって暗号化され電子署名となる。

例えば図 21 の例では、sha256 ハッシュ関数でフィンガープリントを計算し、RSA で暗号化を行っている。電子署名を発行者の公開鍵で復号化し、フィンガープリントを比較すれば、証明書の改変が無いことを確認できる。これによって、所有者の識別名と公開鍵の対応を発行者が保証した証明書を意味することとなる。

一方、CA 自身の正当性は別の CA による公開鍵証明書 (中間証明書) を発行する事を繰り返し、その起点となるトラストアンカー (ルート認証局) の正当性証明は何らかの他の手段で与えられるものとする。WWW においては、WWW サーバは TLS プロトコル内でサーバ証明書と中間証明書を提示し、クライアントとなるブラウザには予めトラストアンカーの証明書 (ルート証明書) が埋め込まれている。図 23 にブラウザで証明書の階層構造

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

03:.....

Signature Algorithm: sha256WithRSAEncryption

Issuer: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3

Validity

Not Before: Nov 29 06:27:50 2018 GMT

Not After : Feb 27 06:27:50 2019 GMT

Subject: CN = example.hi.is.i.nagoya-u.ac.jp

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public-Key: (2048 bit)

Modulus:

00:.....

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Key Usage: critical

Digital Signature, Key Encipherment

X509v3 Extended Key Usage:

TLS Web Server Authentication, TLS Web Client Authentication

X509v3 Basic Constraints: critical

CA:FALSE

X509v3 Subject Key Identifier:

F1:.....

X509v3 Authority Key Identifier:

keyid:A8:.....

Authority Information Access:

OCSP - URI:http://ocsp.int-x3.letsencrypt.org

CA Issuers - URI:http://cert.int-x3.letsencrypt.org/

... 中略 ...

Signature Algorithm: sha256WithRSAEncryption

29:.....

図 21: サーバ証明書の例

```

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      0a:.....
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: O = Digital Signature Trust Co., CN = DST Root CA X3
    Validity
      Not Before: Mar 17 16:40:46 2016 GMT
      Not After : Mar 17 16:40:46 2021 GMT
    Subject: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:.....
      Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Basic Constraints: critical
        CA:TRUE, pathlen:0
      X509v3 Key Usage: critical
        Digital Signature, Certificate Sign, CRL Sign
      Authority Information Access:
        OCSP - URI:http://isrg.trustid.ocsp.identrust.com
        CA Issuers - URI:http://apps.identrust.com/roots/dstrootcax3.p7c
      X509v3 Authority Key Identifier:
        keyid:C4:.....
      X509v3 Certificate Policies:
        Policy: 2.23.140.1.2.1
        Policy: 1.3.6.1.4.1.44947.1.1.1
        CPS: http://cps.root-x1.letsencrypt.org
      X509v3 CRL Distribution Points:
        Full Name:
          URI:http://crl.identrust.com/DSTROOTCAX3CRL.crl
      X509v3 Subject Key Identifier:
        A8:.....
    Signature Algorithm: sha256WithRSAEncryption
      dd:.....

```

図 22: 中間証明書 (Let's Encrypt) の例



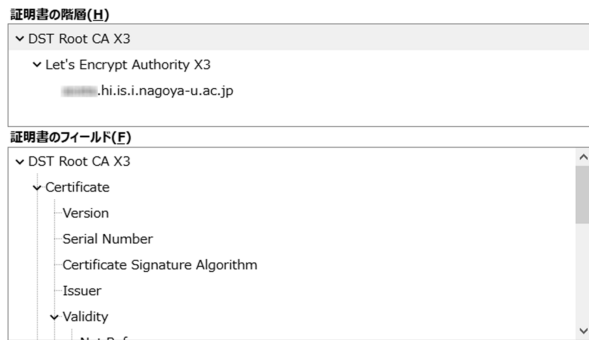


図 23: 証明書の階層構造

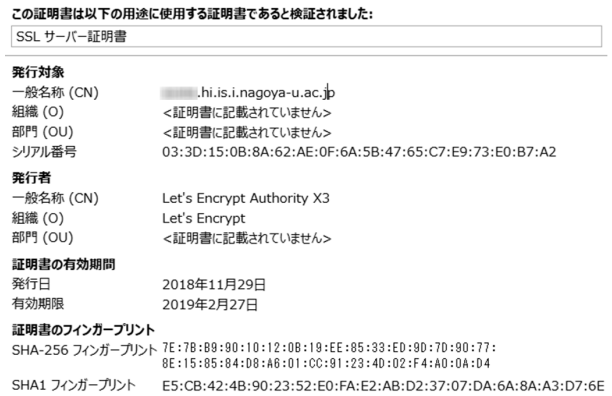


図 24: DV 証明書



図 25: OV 証明書 (Amazon)



図 26: EV 証明書 (住友三井銀行)

(証明書チェーン)を表示した例を示す。また、図 23 における中間証明書を図 22 に示す。

CA の発行するサーバ証明書には、DV (Domain Validation) , OV (Organization Validation), EV (Extended Validation) 証明書の三種類が存在する。ドメイン<sup>25</sup>の認証のみを行う DV 証明書と運営者の実在性審査を行う OV 証明書、厳格に運営者の審査を行う EV 証明書である。銀行等の安全性を強く求められるサーバでは EV 証明書が用いられ、ブラウザの表示は運営者名に変化し、緑色に表示されることで信頼性が保証されていることを明示する。図 24, 25, 26 に DV, OV, EV 証明書をブラウザ (firefox) で確認した例を示す。

PKI は、Subject の情報と公開鍵の対応を保証するための仕組みであり、その有効性は WWW に限定されず、公開鍵暗号を利用するあらゆる機会でも有用である。このため、日本政府の政府認証基盤 (GPKI) 等の公的認証基盤も整備されつつある<sup>26</sup>。

#### (4) サーバ証明書の作成

サーバ証明書の作成手順は以下のとおりである。

1. 秘密鍵の作成
2. 証明書署名要求 CSR の生成
3. CA に CSR を送信し、サーバ証明書を生成

ステップ3で本来は第三者の CA による署名によってサーバ証明書を得るのが一般的な手続きであるが、利用環境によっては、社内の私的 CA による証明書や自分自身による自己署名証明書でも問題ない状況も有りうる。本実験では CSR を自分の秘密鍵で署名した“自己署名サーバ証明書”を作成し、https 通信に利用するものとする。

<sup>25</sup>CSR の CN (Common Name) に対応する。

<sup>26</sup>残念ながら、GPKI の CA である ApplicationCA2 Root は手続き上の不備により、Mozilla Foundation の Firefox ブラウザ等の一部ブラウザのルート認証局から除かれてしまったため、2018 年初頭以降 https 接続ではセキュリティ例外を引き起こすこととなった。

表 14: openssl コマンドの利用方法

サブコマンド	オプション	説明
genrsa [numbits]	-aes128	[numbits 長] RSA 秘密鍵の生成 128bits AES-CBC を用いた共通鍵暗号方式により出力を暗号化
rsa		RSA 鍵の処理
req	-utf8 -new	PKCS#10 X.509 CSR (Certificate Signing Request) の処理 入力文字コードは UTF8 (デフォルト ASCII) 新規リクエスト
x509	-req -signkey file	X.509 証明書データの処理 入力は CSR, 署名し証明書を出力 秘密鍵 file で自己署名証明書を作成
rsa or req or x509 任意	-text -noout -in file -help	鍵 / CSR / 証明書ファイル file の内容をテキスト形式で表示 サブコマンドのヘルプ出力

#### (4) openssl コマンド

openssl コマンドの書式を以下に示す。

openssl サブコマンド [オプション]
------------------------

表 14 に openssl コマンドの利用方法を示す。openssl コマンドの操作対象として、秘密鍵／公開鍵、CSR、X.509 証明書の 3 種類に関しては、それぞれサブコマンド genrsa/rsa, req, x509 サブコマンドを利用する。一方で、生成されたデータの内容とは無関係に、データを保存する形式に、DER 形式と PEM 形式の二種類の形式が存在する。DER 形式はデータをバイナリ形式のまま保存し、PEM 形式では、BASE64 エンコードによって、テキスト化した後保存する形式である。従って、ファイル名の拡張子には、内容を表す .key, .csr, .crt が利用されることもあれば、.pem, .der が利用されることもあり、ファイル内容の確認のためには、openssl コマンドによる確認が必用となることもある。

SSH, TLS はもちろん、すべての暗号化ソフトウェアは幾度も脆弱性の発見・修正を繰り返している。また TLS ではプロトコルバージョンが更新され、古いバージョンのプロトコルは避けることが推奨されている。このように、暗号化技術に関しては日進月歩であるため、最新情報に留意していることが要求され、サーバを放置しておくことは許されない。特に、TLS を利用する他アプリケーションに関しては、プロトコルバージョンを明示して設定することも有り注意が必要である。

#### 4.5.3 SELinux

SELinux (Security-Enhanced Linux) は米国 NSA (National Security Agency, 国家安全保障局) により開発されたカーネルのセキュリティモジュールである。従来 Linux では、4.7 節で述べる UID/GID とファイルパーミッションによるアクセス制御を利用してきた。しかし、SELinux の導入により、これと比較して、はるかに強力なセキュリティ性能を保証することができる。

SELinux の動作モードには、enforcing (強制実行), permissive (警告のみ), disabled (停止) の三種類があり、設定ファイル /etc/sysconfig/selinux で指定することができる。また、現在の状態の確認・設定は getenforce, setenforce コマンドで確認・設定可能である。

SELinux はセキュリティ性能を大きく向上させるため、enforcing モードで実行することが望ましいが、httpd 等のサービスの設定が煩雑となるため、本実験では permissive モードで実行することとする。

## 4.6 パッケージ管理

オープンソースプログラムを配付する際等に、複数のソースファイルをそのまま配付するのは面倒であるため、tar コマンドでファイルを一つにまとめ（アーカイブし）、さらに gzip で圧縮することが良く行われており、tar ball と呼ばれている。Linux の初期に、Slackware 等では、ディストリビューションのバイナリファイルを tar ball で配付していた。しかし、tar ball は単にファイルをアーカイブしただけであり、それ以上のメタ情報は含んでいないため、ソフトウェアの継続管理には不向きである。Red Hat, Debian 等のディストリビューションにおいては、それぞれ独自のパッケージと呼ばれる形式でソフトウェアを統合管理している。RHEL, CentOS, OpenSUSE 等では RPM (RPM Package Manager) 形式、Debian, Ubuntu 等では dpkg (Debian package) 形式を採用している。これらのパッケージは格納ファイルのソフトウェア名称やバージョン、ビルド年月日、インストールパス等のメタ情報を含み、rpm/dpkg 等の管理コマンドでインストール／アンインストールが容易に実行できる。このように、Linux ディストリビューションはパッケージの集合で構成されており、一度インストールが終了してシステムが起動した後であれば、対応するパッケージさえ入手すれば、簡単に機能を拡張することが可能となる。

rpm コマンドは、コンピュータ内に存在するパッケージを管理し、データベースに記録するコマンドである。しかし、CentOS, RHEL 等では長期のサポート期間にわたって、ソフトウェアのバクフィクス／セキュリティフィックスが行われ、アップデートパッケージが配付されている。従って、ネットワーク経由でパッケージ配布サイトのアップデートファイルを確認し、アップデート作業を行ったり、あるいは新しいパッケージを検索してインストールしたりする機能が必要となる。この作業を行うのが yum コマンドである。yum は、パッケージの検索、情報表示、インストール、アンインストール、アップデート等の機能を有している。yum がチェックを行うパッケージ配布サイトの情報を格納したファイルはリポジトリと呼ばれ、/etc/yum.repos.d に格納されている。RHEL/CentOS では、ディストリビューションの公式パッケージ配布サイトのリポジトリは標準で設定されているが、それ以外のリポジトリを追加することで、yum がチェックするサイト（ソフトウェア）を増やすことが可能となる。yum はインストール（アンインストール）時に、パッケージ間の依存性（必要とするライブラリ等）を考慮し、追加で必要（不必要）となるパッケージを追加インストール（アンインストール）する機能を有している。従って、リポジトリに含まれるパッケージに関しては、rpm ではなく yum を利用してインストール（アンインストール）することが望ましい。

セキュリティ対策のためには、インストール済みパッケージのアップデート yum update は定期的に行うことが推奨されるが、Linux の自動実行機能 cron を利用して yum update の定期実行を行うパッケージ yum-cron が存在し、yum install yum-cron でインストールが行われる。

## 4.7 ユーザ管理

Linux システムのユーザは一意のユーザ ID (UID) と第一所属グループ（プライマリグループ）を示すグループ ID (GID) を持つ。新規ユーザの追加は useradd コマンドに適切なオプションを付けて実行することで可能である。これによって、ユーザ管理ファイル /etc/passwd に、UID, GID, ユーザ名, ホームディレクトリパス, ログインシェル, GECOS 情報（フルネーム等の情報）を備えた行が追加され、ユーザのホームディレクトリが生成される。ホームディレクトリ作成時の雛形は /etc/skel であるため、ここに必要な各種初期設定ファイルを用意しておく必要がある。また、GID とグループ名の対応は /etc/group で管理される。各ユーザのパスワードは SHA-512 ハッシュ関数で暗号化され、/etc/shadow に格納される。

上記のローカルユーザ以外にも、ネットワーク経由での集中情報管理を可能とする様々な仕組み、例えば NIS, LDAP や様々な認証サーバ等が存在する。しかし、これらに関しては、本実験と無関係であり、これ以上は触れない。

## 4.8 プロセス管理

OS 上で動作するすべてのプロセスは、所有者（実行者）の UID が与えられ、また PID 番号と呼ばれる番号で管理される。システム管理者は、システムサービスを提供しているデーモンプロセスや一般ユーザの実行するプロセスが Linux システムに与える負荷の状況を監視し、必用ならば介入するために、プロセスの情報を知る必要がある。

以下にプロセス監視・管理を行うコマンドの書式を示す.

- (a) プロセス情報の表示 `ps -ef`
- (b) シグナルの送出 `kill -シグナル番号 PID 番号`
- (c) タスクの連続表示 `top`

一般ユーザにおいても、自分の所有するプロセスに限って制御することができるため、プログラム暴走時等に対応することが可能である.

## 4.9 附録

### (A) シェルの主要な機能

シェルは以下に列挙したような様々な機能を有しており、使いこなすことで、CLI を非常に快適にすることができる. 各機能・利用方法の詳細は各自で学習してもらいたい.

1. シェル変数, 環境変数の利用
2. コマンド履歴
3. (コマンド, ファイル名) 補完
4. ファイル名展開, コマンド置換
5. パイプ, リダイレクト (標準入力, 標準出力, 標準エラー出力)
6. エイリアス

### (B) Linux コマンド一覧

以下に Linux で最も頻繁に利用されるコマンドの一部を列挙する.

コマンド	よく利用されるオプション	説明
man	-f, -S	オンラインマニュアルの表示
ls	-l, -a, -t	ディレクトリの内容をリスト表示
cd		作業ディレクトリの変更
pwd		作業ディレクトリのフルパス名を表示
cp	-r	ファイルやディレクトリのコピー
mv		ファイルの移動/名前の変更
rm	-r, -f	ファイルやディレクトリの削除
mkdir		ディレクトリを作成
chmod	-R	ファイルのモードビット変更
less		テキストファイル閲覧用ページャー
cat		テキストファイルの連結表示
diff		テキストファイル間の相違を出力
head		ファイルの最初の部分を出力
tail		ファイルの末尾の部分を出力
egrep		パターンにマッチする行を表示
tar	-c, -x, -t, -f, -v	アーカイブの作成・抽出
tee		標準入力を標準出力と同時にファイル出力

## (C) man コマンド実行時のページャー (less) の操作キー

man コマンドにおいて、ページャ操作を行うキーを以下に示す（詳細は `man less` 参照）。

キー	動作
SPACE キー	前方に 1 ページ分進む
b	後方に 1 ページ分戻る
d	前方に半ページ分進む
u	後方に半ページ分戻る
G	ファイルの最終行に移動
g	ファイルの 1 行目に移動
/文字列	前方にある文字列を検索
?文字列	後方にある文字列を検索
n	前回の検索を繰り返す
N	前回の検索を逆方向に行う
q	man (less) を終了する

## (D) script コマンドの利用方法

script コマンドは、端末に表示される内容の写しをファイルに保存するコマンドである。以下の様に実行され、`exit` 入力により終了するまで、すべてをファイルに保存する。ログを保存するファイル名 `filename` を指定しないと、常にファイル `typescript` を上書きするため注意が必要である。

```
script [filename]
```

プログラミングの授業で、改行文字やタブ文字を利用した経験があると思うが、文字コードには画面表示を制御するための非表示文字、または制御文字が存在する。また複数文字から構成されるエスケープシーケンスが存在し、man コマンドの表示制御やカラー表示にも利用されている<sup>27</sup>。

script コマンドは、端末に出力される制御文字やカラー表示のためのエスケープシーケンス等、表示を制御するための多くの情報もそのままファイルに保存する。このため、レポートに利用する前に、不要な制御文字等を取り除く必要がある。以下のコマンドによって、多くの制御シーケンスを取り除くことができる。

```
cat ファイル名 | sed -r "s/\x1B\[([0-9]{1,2}([;[0-9]{1,2})*)?m//g" | col > 新規ファイル名
```

しかし、上記コマンドもすべての制御文字を取り除くわけではないため、vi コマンドによるエディタ編集作業時等、レポートに記載が不要な作業時には、script コマンドを終了しておく方が望ましい。また、削除文字 (BackSpace) が取り除かれることによって、消したはずの誤入力が復活してしまうため、注意が必要である。

<sup>27</sup>man で利用されている roff 文書清書システムは、4.1.1 節で述べた UNIX 誕生時の社内プロジェクトで開発され、現在でも利用・開発され続けている

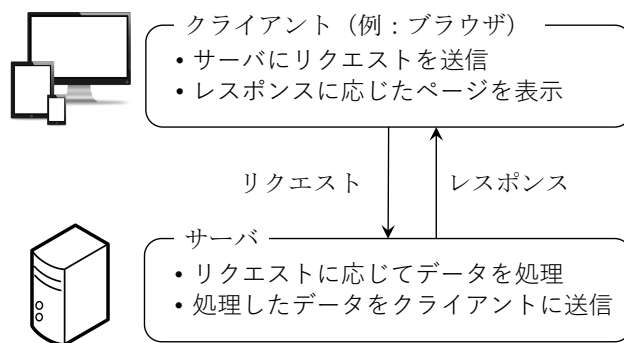


図 27: WEB アプリケーションの構成図。

## 5 WWW サーバによる WEB アプリケーションプログラミング

WEB 上で見かける WEB ページは、WEB アプリケーションとして構築されている。WEB 検索エンジンは、入力されたキーワードを処理し検索にマッチする WEB ページを表示する。ブログサービスは、記入した記事をサーバに保存したり、記事を読むためにブラウザに表示する。WEB アプリケーションは、サーバ・クライアントモデルで設計されている (図 27)。クライアントは、ユーザにデータを表示しサーバにリクエストを送信する役割を担う。サーバは、クライアントからのリクエストに応じてデータを処理し、クライアントにレスポンスを返答する。

WEB 検索エンジンを例にとると、まずクライアントは WEB 検索エンジンに検索のためのページの表示をリクエストする。WEB 検索エンジンは受け取ったリクエストに応じて、検索窓を表示するためのデータをクライアントへレスポンスとして返答する。クライアントは、レスポンスを受け取り、受け取ったデータに応じたページ (検索窓のあるページ) を表示する。ユーザは、クライアントに表示されたページの検索窓にキーワードを入力し、検索ボタンをクリックする。クライアントは、検索窓に入力されたキーワードを検索クエリとしてリクエストに含め、サーバに送信する。サーバは受け取ったリクエストから、WEB 検索をするタスクであることを知り、そのクエリがリクエストに含まれているキーワードであることを知る。サーバはこれらの情報から、WEB ページを検索し、キーワードに該当する WEB ページを探し出す。サーバは、見つけ出した WEB ページを適当な並び順で表示するためにデータを整理し、レスポンスとしてクライアントへ返答する。クライアントは、受け取ったレスポンスを解析し、キーワードにマッチした WEB ページのリストを表示する。ユーザは、表示されたリストから興味のある WEB ページへのリンクをクリックし、その WEB ページへアクセスする。アクセス先のページでも同様のプロセスでデータにアクセスする。

クライアントがサーバから得たレスポンスを表示するにはどうしたら良いだろうか？

WEB アプリケーション毎に表示するシステムを作っていたのではキリがない。そこで、WEB アプリケーションのページ表示用のマークアップ言語 HTML (Hyper Text Markup Language)<sup>28</sup>が W3C (World Wide Web Consortium)<sup>29</sup>により標準規格として設計された。WEB ブラウザはこの標準規格に準拠して、HTML で決められた方法で WEB ページを表示する。HTML はデータの構造を記述する方法に過ぎず、より見やすい見た目にするためには CSS (Cascading Style Sheet) を用いてデザインを設計する<sup>30</sup>。5.1 節で、HTML と CSS について紹介する。

サーバはどのようにしてクライアントからのリクエストを受け取り、レスポンスを返したら良いだろうか？

クライアントがサーバにリクエストを送ったり、サーバがレスポンスを返答したりする際に、どう連携するかを予め決めておかないとうまく連携が取れずアプリケーションの構築が困難になる。そのために、どのようなインタフェースを持つべきかの標準規格が設計されている。広く使われているものに、CGI (Common Gateway Interface)<sup>31</sup> があり、さまざまなプログラミング言語で CGI を記述するためのフレームワークが用意されている。WSGI<sup>32</sup>はプログラミング言語 Python におけるインターフェースの標準仕様である。5.2 節で、CGI と WSGI について紹介する。

<sup>28</sup><https://www.w3.org/html/>

<sup>29</sup><https://www.w3.org/>

<sup>30</sup>JavaScript を用いるとクライアント上で動作するページを構築可能であるが、本実験の範囲を超えるためここでは扱わない。

<sup>31</sup><https://tools.ietf.org/html/rfc3875>

<sup>32</sup><https://www.python.org/dev/peps/pep-0333/>



```

<html>
  <head>
    <title>食べ物の色</title>
  </head>
  <body>
    りんごは赤い。
  </body>
</html>

```

図 28: HTML の基本構造.

```

食べ物の色

```

りんごは赤い.

図 29: ブラウザ表示.

## WEB アプリケーション上でデータはどのように管理・アクセスしたら良いだろうか？

プログラムを実行する際、普段ならファイルへの読み書きで、データを読み出したり書き出したりするだろう。例えば、WEB アプリケーションの場合、複数のユーザが同時にアクセスしたり、データを更新する可能性がある。こういった場合にデータの不整合が起きやすくなる。複数の更新が同時に発生し、一方の更新は反映されるがもう一方の更新は反映されない、などが考えられる。データベース（特に関係データベース）はこのような不整合が起きないように設計されたデータ管理システムである<sup>33</sup>。データベースを利用してデータの操作（挿入・削除・検索）を行う際には SQL (Structured Query Language)<sup>34</sup> を用いる。SQL は関係データベースに対する操作言語である。5.3 節で、SQL について紹介する。

## 5.1 HTML と CSS

本節では HTML と CSS について簡単に紹介する。なお、HTML の要素や CSS のプロパティはかなりの種類数があるため、ここでは主要なものや使い方について述べるに留める。HTML の要素や CSS のプロパティについては、各自で調べられたい。w3schools.com (<https://www.w3schools.com/>) がリファレンスとして有用であるので、ぜひ参照されたい。

### 5.1.1 HTML の基本構造

HTML とは、マークアップ言語の一種である。マークアップ言語とは、データの構造を記述するための形式言語である。HTML は構造化された文書であり、WEB ブラウザはその構造に従ってデータを表示する。例えば、「<strong> 強調 </strong>」という記述は、「強調」という文字列を strong な方法で表示することを意味する。「strong な方法」は CSS やブラウザのデフォルト設定などのスタイルによって決まっている。

HTML では、タグ (tag) と呼ばれるマークアップの目印を使用し、データの構造を表す。HTML で必須のタグとして、<html> がある。これは、HTML 文書であることを宣言するタグである。タグは、< と > によって囲まれた文字列を指す。タグには、開始タグ (start tag) と終了タグ (end tag) があり、開始タグと終了タグで挟んだ文字列にタグで意味をもたせる。開始タグは “<x>” のようにタグの名前 x を “<” と “>” に挟んで書き、終了タグは “</x>” のように、“</” と “>” に挟んで書く。よく使うタグのひとつに p がある。これは、パラグラフを表すタグで、間に挟まれた文字列がひとつのパラグラフであることを示す。p タグは例えば、「<p>HTML は Hyper Text Markup Language の略です。</p>」のように使う。タグの中にタグを含めることも可能である。ただし、<x><y>...</x></y> のように開始タグと終了タグを互い違いに記述してはならない。必ず、<x><y>...</y></x> のように囲んでいるタグ（この場合は x）の開始タグと終了タグの間に、囲まれているタグ（この場合は y）の開始タグと終了タグが含まれていなければならない。このように、HTML においてタグは入れ子構造となっており、木構造として解釈することができる。

各タグには属性 (attribute) を付与することができる。属性は、タグに必要な情報だったり、スタイルを決める際に必要な情報だったりする。属性は、開始タグのタグ名の横にスペース区切りで属性・値のペアで記述する。例えば、リンクを表す a タグを例にとると、<a href="http://www.nagoya-u.ac.jp/"> のようになる。

<sup>33</sup>詳しくはデータベースについての講義や文献を参考にされたい。

<sup>34</sup>国際標準規格 ISO/IEC 9075



表 15: 主要な HTML タグ

タグ	説明	タグ	説明
p	パラグラフ	table	表
br	改行	thead	表のヘッダー
hr	コンテキストの切り替え (横棒)	tbody	表の本体
div	セクション	tr	表の行
span	セクション	th	表のヘッダセル
h $x$	見出し ( $x$ には 1...6 の数字が入る)	td	表のセル
img	画像	form	フォーム
a	ハイパーリンク	input	フォームへの入力
ul	順序番号なしリスト	select	ドロップダウンメニュー
ol	順序番号つきリスト	option	ドロップダウンメニューの要素
li	リストの要素		

この例では、リンク先の URL が “<http://www.nagoya-u.ac.jp/>” であることを表す。href という属性に対して “<http://www.nagoya-u.ac.jp/>” という値を割り当てる。各タグについて、任意の数の属性をスペース区切りで付与することができる。

HTML で WEB ページを書く際に、ブラウザが理解するための基本構造が決まっている。基本構造は、図 28 に示すような構造である。まず、HTML であることを宣言するために、html タグを書く。html タグで囲まれた文字列が HTML 文書と認識される。html タグの子要素として、head タグと body タグがある。head タグは、HTML 文書を表示する上で必要な情報やメタデータを記述する要素である。図 28 の例では、title タグが挿入されており、この HTML 文書のタイトルを表す。後述する CSS はこの head タグ内に記述する。body タグは HTML 文書の本体 (body) で、このタグの内側に文書として記述したい内容を記述する。

### 5.1.2 主要なタグ

表 15 に HTML でデータを記述するときによく利用されるタグを示す<sup>35</sup>。各タグについてどのように表示されるかを試してほしい。br タグや hr タグは一般に間に挟む文字列が存在しない (空文字列を挟むとも考えられる)。このようなタグの使い方をするときは、開始タグと終了タグと一緒に `<br />` のように書くことができる。div タグや span タグはデータの区画を決めるタグで、区画に意味をもたせたいときに使う。h1 タグ、h2 タグや h6 タグは見出しを表すタグで、以降のデータの見出しを表現できる。h の次の数字はデータの階層に対応する。例えば、第一章なら h1 タグ、第一章一節なら h2 タグのように利用できる。img タグは画像を表すタグで、src 属性を必要とする。src 属性に画像への参照 (URL やファイルパス) を値として入れることで画像を表す。img タグも br タグや hr タグのように開始タグと終了タグを同時に表記する事が多い。上記でも示したリンクを表す a タグは、`<a href="http://www.nagoya-u.ac.jp/">名古屋大学</a>` のように書く。タグで挟まれた文字列はアンカーテキストと呼ばれ、ブラウザ上で表示される内容である。ブラウザ上でアンカーテキストをクリックすることで、リンク先の WEB ページを表示する。

ul タグと table タグはリストや表を書くためのタグであり、データを可視化するのに向いている。図 30 は、ul タグと table タグを使ったリストと表の書き方の例である。ul タグは、li タグと一緒に使われる。ul タグが順序番号なしのリストであることを宣言し、li タグでリストの要素を表現する。図に示すように、li タグの中身が黒点の横に書かれ、上から順に列挙されている。ol タグは ul タグの姉妹タグでリストに順序番号をつけるタイプのリストである。どの様になるかは各々試されたい。table タグは、thead タグと tbody タグを子要素に持つ。それぞれ、表のヘッダと中身に対応する。thead タグは更に子要素として tr タグをもつ。tr タグはthead タグとtbody タグに共通で、表の行を表す。tr タグは子要素に、th タグあるいは td タグをもつ。th タグは表のヘッダセルを表し、td タグは表のセルを表す。図に示すように、th タグを使うと太字で表示され、ヘッダであることが強調される。td タグは、そのまま表示している。図では、表の形が見えるように、table タグに border 属性を付与している。

<sup>35</sup> これら以外にも様々なタグがあるので、各自で調べられたい。

```

<html>
... 中略 ...
<body>
  <ul>
    <li>りんごは赤い. </li>
    ... 中略 ...
  </ul>
  <hr />
  <table border='1px black solid'>
    <thead>
      <tr>
        <th>食べ物</th>
        <th>色</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>りんご</td>
        <td>赤</td>
      </tr>
      ... 中略 ...
    </tbody>
  </table>
</body>
</html>

```

食べ物の色	
<ul style="list-style-type: none"> <li>りんごは赤い.</li> <li>ぶどうは紫色.</li> <li>みかんは橙色.</li> </ul>	
食べ物	色
りんご	赤
ぶどう	紫
みかん	橙

図 30: リストと表. 左は HTML で右はブラウザでの表示である.

HTML において、サーバとやり取りをするために form タグがしばしば用いられる<sup>36</sup>. form タグの使い方の例を図 31 に示す. form タグは action 属性と method 属性をもつ. action 属性はサーバの URL を値に持ち、method 属性は form に含まれるデータを送信するための HTTP メソッド（主に、GET か POST）を指定する. form タグ内には、サーバに送信したいデータを key-value のペアで与える. そのために、input タグを用いて送信するデータを入力できるフォームを作る. データを受け取る側のサーバでは、name 属性に指定された名前に基づいてデータにアクセスする. input タグには type 属性があり、入力する形式によって表示を変えることができる. 例えば、文字列を入力にしたい場合には、text を指定する. 他にも、checkbox でチェックボックスを作れたり、radio でラジオボタンを作ることができる. フォームに入力した情報を送信するために、送信ボタンを作る. この場合には、input タグの type 属性に submit を値として入れる. フォームからデータを送信したいが、そのデータをブラウザ上で表示したくない場合がある. その場合には、type 属性に hidden を指定する. 図の例では、user が 'John Smith' であるという情報をフォームに含めているが、ブラウザ上では見ることができない. これにより、送信ボタンが作られ、ボタンをクリックすることでフォームへの入力をサーバに送信することができる. 図の例では color.cgi にフォームを送信することになるが、実際に color.cgi がなければエラーが返ることになる.

### 5.1.3 CSS

これまでのところで、HTML を使ってデータを表現できるようになった. データをより視覚的に見せるためには、色や大きさなどを工夫することが大事である. CSS は、簡素な HTML に色や大きさなどの調整を施す役割を果たす. CSS は HTML の head タグ内に style タグ（type 属性に text/css を指定）を使って記述する. 外部ファイルとして記述する方法もあるがここでは割愛する.

図 32 に CSS を記述した HTML の例とそのブラウザ表示を示す. head タグ内の style タグ以下が CSS 部分である. CSS を HTML 内に書く場合には、慣習的に <!-- と --> の間に記述する. CSS では、要素をタグ名、クラス名、ID のどれかで指定し、スタイルを記述する. クラス名や ID は各タグに任意につけられる属性 class と id の値に対応する. 要素を指定するときの指定の仕方は、それぞれ異なる. タグ名はそのまま書き（例：図中の

<sup>36</sup>a タグや JavaScript でも実現できる、JavaScript で記述するほうが返り値の表示などで柔軟な表示ができるが、ここではまず基礎的な form の使い方を知ってもらいたい.

```

<html>
<head>
  <title>食べ物の色</title>
</head>
<body>
  <form action="color.cgi" method="GET">
    色を知りたい食べ物は？
    <input type="text" name="food" />
    <input type="submit" value="Submit" />
    <input type="hidden" name="user" value="John Smith" />
  </form>
</body>
</html>

```

図 31: フォーム。左は HTML で右はブラウザでの表示である。

table), クラス名は先頭に. (ドット) をつけて書き (例: 図中の .red), ID は先頭に # をつけて書く (例: 図中の #food\_table). CSS では, 「指定要素 { プロパティ名: 値; ... }」のようにスタイルを記述する. これにより, 指定された要素にプロパティと値で定義されるスタイルが適用される. 例えば, 図中の 「.red { color: red; }」は, red クラスの要素のテキストを赤色にすることを意味する. 実際に, span タグの red クラスのテキストが赤くなっていることが図からわかるだろう. 他にも, table タグに枠線をつけたり, tbody 要素の偶数番目 (even) の要素の背景色を灰色に変えたり, といったことが記述してある. 指定要素の記述に > があるが, これは指定要素の子要素に対して指定したい場合に使う. 「tbody > tr」では tbody タグ以下にある tr タグを指定している. 同じ tr タグでも thead タグ以下の tr タグには適用されないことに注意したい. さらに高度な記述として, 指定タグに対する疑似クラスを: (コロン) で指定できる. 「tbody > tr:nth-child(even)」では, 偶数番目の要素を指定している. even の代わりに 1 や 2 などの数字を入れれば, 任意の場所の要素を指定できる. これらの記述方法は重ねて使うことができる. 「#food\_table > thead > tr > th」がどの要素を指定しているか, もう分かるでしょう. 指定できるプロパティは要素ごとに多少の違いがあるので, 各々調べられたい.

## 5.2 CGI と WSGI

CGI (Common Gateway Interface) は図 27 に示したサーバ側のプログラムの標準的な機構である. CGI を用いることで, 動的な WEB ページ を作成することができる. CGI はあくまで標準的な機構を決めているため, 多くのプログラミング言語で CGI を利用可能である. WSGI (Web Server Gateway Interface) はプログラミング言語 Python において, WEB サーバとアプリケーションを接続するために標準化された機構である. 本実験では, WSGI を主に使うため, CGI の詳しい説明は省略する.

### 5.2.1 動的な WEB ページと静的な WEB ページ

動的な WEB ページとは, ユーザがアクセスした時点で初めてコンテンツが決まる WEB ページのことを指す. 例えば, アクセスカウンターやブログなどが動的な WEB ページに該当する. これらは, ユーザがアクセスした時点でアクセス数を表示したり, 最新の記事を表示したりとアクセスする都度表示されるコンテンツが変わる. 結果的に生成されるコンテンツが同じでも, アクセスするたびにコンテンツが作られていることがポイントである.

一方で, 静的な WEB ページとは, コンテンツが変わる仕組みを持たない WEB ページのことである. 前節で扱った HTML を用いた WEB ページはこれにあたる. HTML 文書が書き換わらない限り, 同じ WEB ページにアクセスすると同じコンテンツが表示される.

### 5.2.2 WSGI

基本的なアプリケーションを作成するためには, wsgiref パッケージの simple\_server を使うのが簡単である. simple\_server では make\_server 関数で簡単に WEB サーバを立てることができる. make\_server 関数は, ホス

```

<html>
<head>
<title>食べ物の色</title>
<style type="text/css">
<!--
table {
border : 1px black solid;
}
tbody > tr:nth-child(even){
background-color: #ddd;
color : black;
}
.red {
color : red;
}
.purple {
color : purple;
}
.orange {
color : orange;
}
#food_table > thead {
background-color : #ffa;
}
#food_table > thead > tr > th {
border-bottom: 1px black dashed;
}
-->
</style>
</head>
<body>
<ul>
<li>りんごは<span class="red">赤い</span>. </li>
... 中略 ...
</ul>
<hr />
<table id="food_table">
... 中略 ...
</table>
</body>
</html>

```

食べ物の色

- りんごは赤い.
- ぶどうは紫色.
- みかんは橙色.

食べ物色	
りんご	赤
ぶどう	紫
みかん	橙

図 32: CSS の例. 左は HTML で右はブラウザでの表示である.

ト、ポート、アプリケーションを引数にとり、指定されたホストとポートで待機し、アクセスが来たらアプリケーション（関数）を呼び出す。サーバの起動には、上記を記述した Python ファイル（例えば、sample.py）を作成し、python sample.py で実行する。このサーバにブラウザからアクセスするには、<http://ホスト:ポート> にアクセスする。ホストをから文字列（"）にした場合、localhost でアクセスできる。

下記にシンプルな例を示す。この例は、アクセスされたら 'Hello World' の文字列を返す例である。

```

def application(environ, start_response):
    start_response('200 OK', [('Content-type', 'text/plain')])
    return ['Hello World']

from wsgiref import simple_server
if __name__ == '__main__':
    server = simple_server.make_server('', 8080, application)
    server.serve_forever()

```

前節で説明した form タグでは、様々な入力をサーバに渡すことができる。この入力をサーバで受け取るために、

cgi パッケージを使って、値を取得する。cgi.FieldStorage 関数によってサーバに送られてくるフォーム (form) を取得する。form.getvalue 関数を用いることで、指定した名前をもつフィールドの入力値を取得する。この関数の第一引数は入力名で、第二引数は入力値が空だった場合のデフォルト値である、下記にフォームからの入力をそのまま返す例を示す。入力が空だった場合に「りんご」を返す。

```
import cgi

def application(environ, start_response):
    form = cgi.FieldStorage(environ=environ, keep_blank_values=True)
    food = form.getvalue('food', 'りんご')
    start_response('200 OK', [('Content-type', 'text/plain')])
    return [food]

from wsgiref import simple_server
if __name__ == '__main__':
    server = simple_server.make_server('', 8080, application)
    server.serve_forever()
```

ここまででサーバがどのように値を返すかを説明した。これらの応用で、HTML 文書を動的に作成し、返すことも容易に実現できるだろう。その際に注意したいのが、start\_response に与えている Content-type の値である。これまでの例は単なる文字列を返していたが、HTML 文書を返す場合にはそれが HTML 文書だと受け取り手 (アプリケーション) にわかってもらったほうが都合が良い<sup>37</sup>。Content-type とペアになっている text/plain はそれが平文の文字列であることを表している。これを HTML 文書であることを表すには、text/html にすればよい。

下記に HTML 文書を生成し返すサーバプログラムの例を示す。入力として食べ物の名前を受け取ると、その色を返す。html という変数に HTML 文書を構築していき、最後に return 文で返している。

```
import cgi

def application(environ, start_response):
    form = cgi.FieldStorage(environ=environ, keep_blank_values=True)
    food = form.getvalue('food', 'りんご')
    html = '<html>\n' \
        ... (中略) ... \
        '<p>%s の色は' % food
    if food == 'バナナ':
        html += '黄色</p>\n'
    ... (中略) ...
    else:
        html += '赤色</p>\n'
    ... (中略) ...
    html += '</html>'
    start_response('200 OK', [('Content-type', 'text/html')])
    return [html]

from wsgiref import simple_server
if __name__ == '__main__':
    server = simple_server.make_server('', 8080, application)
    server.serve_forever()
```

## 5.3 SQL

データを管理する方法には、データベースを使う方法と使わない方法がある。後者は主にファイルでデータを管理する方法である。ファイルでの管理は基本的な方法であるが、WEB アプリケーションで利用するデータを管理するには向かない場合がある。例えば、ユーザが商品を購入する電子商取引 (Amazon など) のアプリケーションを考えてみる。当然複数のユーザが同時にアプリケーションにアクセスすることになる。商品を購入した記録をとっておくために、ファイルにデータを書き込む。単一のファイルに複数のユーザの情報がある場合、同時に

<sup>37</sup>電話で番号を通知してもらったほうが受け取ったときの対応を事前に考えられるのと同様のことだと考えてもらえば良い。



表 16: 関係データベースの例.

(a) 大学データベース					(b) 都道府県別人口: データベース			
名前	略称	所在地	創立 (年)	種別	都道府県名	総人口	人口 (男)	人口 (女)
名古屋大学	名大	愛知県	1871	国立	愛知県	7,539,185	3,771,778	3,767,407
東京大学	東大	東京都	1877	国立	東京都	13,857,664	6,819,000	7,038,664
同志社大学	同志社	京都府	1875	私立	京都府	2,592,553	1,239,036	1,353,517

書き込みが発生することになる。書き込みを行うプログラムのスケジュールによっては、一方の書き込みのみが反映される、などの事態が起こりうる。では、ユーザごとにひとつのファイルとするのはどうか。ユーザ数分だけファイルが増え、ファイルの管理が煩雑になる。数百万を超えるユーザ数を扱おうと思った場合に、相当な工夫をしないと効率的なシステムが作れないことは容易に想像できるだろう。

データベースは、ファイル管理から整合性の管理までデータの管理を担うものである。厳密には、データベース管理システムがこれらの機能を担っている。詳しくは、データベースの講義や文献を参考にされたい。本実験では、代表的なデータベースである関係データベース (Relational Database) をデータベースとして用いることとする<sup>38</sup>。関係データベースは、最も歴史の長いデータベースのひとつであり、WEB アプリケーションのデータ管理でよく使われている。関係データベースは、データを表形式で表現するデータベースであり、表形式のデータにアクセスするために SQL をデータベース操作言語として用いる。

### 5.3.1 リレーション

関係データベースにおいて、表 (リレーション; Relation) は属性付きの行 (厳密にはタプル; tuple) の集合として表現される。行ひとつが単一のデータとして表現される。表 16 に例を示す。表 16(a) は、大学の情報を保有するデータベースである。大学を表現する属性として、名前、略称、所在地、創立、種別を設定した。表 16(b) は、県別の人口のデータベースである。それぞれの県 (各行) について、都道府県名と総人口と男女別の人口がデータとして入っている。

### 5.3.2 SQL

SQL は関係データベースに対して、データの操作・定義を行う言語である。以下では、SQL の基本的な使い方として、(1) 表の定義、(2) 表の操作 (挿入・削除・更新)、(3) 表に対する検索 (選択・射影・結合・集約)、を説明する。なお、これらの使い方は、SQL の一部の使い方であるので、他の使い方について興味がある人は各自で調べられたい。

#### 表の定義: CREATE TABLE 文

表の定義には、CREATE TABLE 文を用いる。表の定義は、表の雛形を作る作業である。表 16 に示した表の一行目を作ることに相当する。CREATE TABLE 文は以下のような構文で記述する<sup>39</sup>。CREATE TABLE の後に、table\_name で表の名前を指定する。その後に、カッコ内に各属性の名前とデータ型を指定する。「attr\_i type\_i」のように、スペース区切りで、属性名とデータ型を記述する。これをカンマ (,) 区切りで書くことで表の属性集合を記述する。データ型には、int (整数値)、float (浮動小数点数値)、text (文字列) など指定する<sup>40</sup>。

```
CREATE TABLE table_name (attr_1 type_1, attr_2 type_2, attr_3 type_3, ...)
```

<sup>38</sup>XML データを扱うための XML データベース (例えば、eXist (<http://exist-db.org>)) やグラフデータを扱うためのグラフデータベース (例えば、Neo4j (<https://neo4j.com/>))、JSON データを扱うための NoSQL データベース (例えば、MongoDB (<https://www.mongodb.com/>)) など存在する。興味がある人は各々調べられたい。

<sup>39</sup>実際には、主キー制約など表に関する付加情報を同時にかけられるのだが、ここでは最も簡単な表の定義方法にとどめる。

<sup>40</sup>サポートしているデータ型は関係データベースシステムに依存する。最近の関係データベースは多くのデータ型をサポートしているので、気になる人は調べられたい。

表 16(a) を定義するには、以下のように書く。表の名前を university とし、それぞれの属性名を name (名前), abbrev (略称), location (所在地), foundation (創立), type (種別) とした。

```
CREATE TABLE university (name text, abbrev text, location text, foundation int, type text)
```

同様に、表 16(b) を定義するには、以下のように書く。表の名前を population とし、それぞれの属性名を prefecture (都道府県名), population (総人口), male\_population (人口 (男)), female\_population (人口 (女)) とした。

```
CREATE TABLE population (prefecture text, population int, male_population int, female_population int),
```

### 表の操作 (挿入) : INSERT 文

表へデータを挿入するには、INSERT 文を用いる。挿入は、定義した表に行を挿入する操作である。INSERT 文は以下のような構文で記述する。INSERT INTO の後に挿入したい表の名前を指定する。その後に、挿入するデータの属性を表の属性から選んで指定する。このときの順番は、表の定義の順番と異なっても構わない<sup>41</sup>。VALUES 以降に、挿入したい値を指定した属性の順番に沿って記述する。このとき、属性の指定順と異なる順序で値を記述してしまうと期待した属性とはことなる属性に値を挿入してしまうことになるので、順番には十分注意されたい。複数行を同時に入力することも可能である。

```
INSERT INTO table_name (attr_1, attr_2, attr_3, ...) VALUES (value_1, value_2, value_3, ...)
```

表 16(a) の二行目を挿入するには、以下のように書く。例示するためにあえて、VALUES で指定しているタプルの属性値の順番を表定義での属性の順番と変えている。text 型を扱う場合には、シングルクォート (') で囲う必要があることに注意されたい。

```
INSERT INTO university (name, abbrev, type, location, foundation)
VALUES ('名古屋大学', '名大', '国立', '愛知県', 1871)
```

### 表の操作 (削除) : DELETE 文

表からデータを削除するには、DELETE 文を用いる。削除は、表から行を削除する操作である。DELETE 文は以下のような構文で記述する。DELETE FROM の後に、行を削除したい表の名前を指定する。WHERE 以降に削除する条件を記述する。これによって、条件にマッチする行を「すべて」削除する。WHERE を含む以降を記述しないとすべての行を削除するので、操作には十分注意されたい。条件の書き方は「表に対する検索」の項目で説明する。

```
DELETE FROM table_name WHERE 条件
```

表 16(a) の二行目を削除するには、以下のように書く。

```
DELETE FROM university WHERE name = '名古屋大学'
```

<sup>41</sup> 挿入するデータの属性の順番が表の定義と同じ場合には、省略できる。その場合は、VALUES 以降に記述する値の順番は表の定義にある属性の順番に従う必要がある。



### 表の操作（更新）：UPDATE 文

表内のデータの値を更新するには、UPDATE 文を用いる。更新は、表の特定のセル（表内のマス）の値を書き換える操作である。表 16(a) の一番下の行の略称「同志社」を「同大」に書き換える操作などが更新の例である。UPDATE 文は以下のような構文で記述する。UPDATE の後に、更新したい表の名前を指定する。SET の後に、更新したい属性名と更新後の値を = で結んで列挙する。WHERE の後に、更新する行の条件を記述する。これによって、条件に合致する行の指定した属性の値が更新される。削除と同様に WHERE 以降を記述しなかった場合にすべての行の値が更新されるので注意されたい。条件の書き方は、「表に対する検索」の項目で説明する。

```
UPDATE table_name SET attr_1 = val_1, attr_2 = val_2, ... WHERE 条件
```

表 16(a) の一番下の行を更新するには、以下のように書く。

```
UPDATE university SET abbrev = '同大' WHERE name = '同志社大学'
```

### 表に対する検索：SELECT 文

データベースに格納されたデータに対して検索するには、SELECT 文を用いる。検索はデータベース内の表から条件に合致するデータを見つけ出すことである。検索に使える演算は、主に選択 (selection)、射影 (projection)、結合 (join)、集約 (aggregation) である。

- 選択：指定した条件に合致した「行」を検索する。
- 射影：指定した属性に対応した「列」を抽出する。
- 結合：二つの表から条件に合致する行同士をつなぎ合わせる。
- 集約：指定した属性で行をまとめ上げる。集約値を計算する関数と一緒に使う。

SELECT 文は以下のような構文で記述する。SELECT の後に射影条件として、結果として出力したい属性リストを記載する。属性リストの代わりに\*を記載すると表のすべての属性を出力する。FROM の後に検索対象となる表のリストを記載する。二つ以上の表を書いた場合に結合演算を行う対象になる<sup>42</sup>。WHERE の後に選択条件として属性に対する比較条件のリストを記載する。比較条件には、<, <=, >=, >, =, != 演算子を用いることができる。結合条件も記述することができる<sup>43</sup>。GROUP BY の後に集約対象としたい属性リストを指定する。例えば、グループごとの行の数を数え上げる時には、COUNT 関数で計算する。グループ内の特定の属性の値の総和を取りたい場合には、SUM 関数を使う。指定する順番に意味があり、先に記述されている属性でグループ化し、各グループ内で後述されている属性でグループ化を行う。

```
SELECT 射影条件  
FROM 表リスト  
WHERE 選択条件  
GROUP BY 集約条件
```

表 16(a) の表から国立大学の大学名と所在地を取り出す検索を考える。この検索は、取り出したい属性は大学の「名前 (name)」と「所在地 (location)」で、条件として「種別 (type)」が「国立」のものを検索する。これを SELECT 文の形で記述すると以下ようになる。

```
SELECT name, location  
FROM university  
WHERE type = '国立'
```

名前	所在地
名古屋大学	愛知県
東京大学	東京都

この検索結果は以下の表になる。

次に、表 16 の二つの表を用いた検索を示す。人口が三百万を超える都道府県にある大学の大学名 (name) と所在地 (location) および総人口 (population) を検索し、総人口の多い順に並べる、これを実現するためには、二つの表の同じ都道府県名を持つ行を結合する必要がある。二つ以上の表を FROM で指定する場合には、その属性にアクセスする際に曖昧さを回避するため、「表名. 属性名」でアクセスする。二つ以上の選択条件を書く際には、AND や OR を使う。並べ替えには ORDER BY を用いて並べ替える属性リストを指定すると同時に、昇順 (ASC) か降順 (DESC) を指定する<sup>44</sup>。

```
SELECT university.name, university.location, population.population
FROM university, population
WHERE university.location = population.prefecture AND population.population >= 3,000,000
ORDER BY population.population DESC
```

この検索結果は以下の表になる。

名前	所在地	総人口
東京大学	東京都	13,857,664
名古屋大学	愛知県	7,539,185

最後に、集約演算を用いた検索の例を示す。この例では、表 16(a) から種別 (type) ごとの大学の数を数え上げる。集約演算と数え上げ関数 (COUNT) を用いて以下のように記述する。集約対象の属性は必ず SELECT の後ろで指定しなければならないことに注意されたい。

```
SELECT type, COUNT(type)
FROM university
GROUP BY type
```

この結果は以下のようになる、集約値を計算する関数の属性は、その関数の名前になるのが一般的である。

種別	COUNT
国立	2
私立	1

集約した値で絞り込みをしたい場合がある。そのような場合には、GROUP BYに加えて、HAVINGを使うことで、集約した値が条件に合致するものだけを検索することができる。例えば、以下のようにすると、表の中で、1回を超える出現をした種別のみを検索することができる。

```
SELECT type, COUNT(type)
FROM university
GROUP BY type
HAVING count(type) > 1
```

この結果は以下のようになる。

<sup>42</sup>結合条件を指定しない場合、各表の行集合の直積（取りうるすべての組合せ）が結合結果となる。

<sup>43</sup>結合条件を記述する方法は他にもあるが、説明が複雑になるため、ここでは WHERE の後を書く方法を紹介する。

<sup>44</sup>デフォルトは昇順なので、属性のみを書くとは昇順になる。

種別	COUNT
国立	2

### 5.3.3 Python からデータベースへアクセス

本実験では、軽量な関係データベースである SQLite<sup>45</sup>を用いる。Python からの関係データベースへのアクセスはおおよそどの関係データベースでも共通である。まず、データベースにコネクションを張り、カーソルオブジェクトを用いて SQL の処理を実行する。処理を実行 (`cur.execute`) したときにそれが SELECT 文なら問い合わせ結果の行の配列が返ってくる。各行について、属性値は SELECT で指定した順番（射影の条件が\*の場合は表定義通りの順番）に並んでいるのでインデックスを使って取得する。

下記に Python でデータベースにアクセスする例を示す。

```
import sqlite3

dbname = 'sample_db'
con = sqlite3.connect(dbname)
cur = con.cursor()

query = 'select name, location from university'
for row in cur.execute(query):
    print('%s は %s にある' % (row[0], row[1]))

cur.close()
con.close()
```

### 5.3.4 データのインポート

データベースを構築する際に、既存のデータを使いたい場合がある。既存のデータをデータベースに入力するには、(1) 5.3.2 の INSERT 文を使う方法と (2) データを一括でインポートする方法がある。前者はすでに説明したように、各行を INSERT 文を生成するプログラムを記述し、入力する方法である。データの一部を予め加工してデータベースに入力したい場合はこの方法がいいだろう<sup>46</sup>。後者の方法は、データベース管理システムに用意されているインポート機能を利用する方法である。利点は、INSERT 文のような整形が必要なく、CSV (Comma-Separated Values) 形式や TSV (Tab-Separated Values) 形式などの標準的なフォーマットをサポートしている点である。

データベース管理システムごとに機能の名前や使い方が異なるため、ここでは SQLite でのやり方を紹介する。

1. `sqlite3` コマンドでデータベースへの対話的インターフェースを起動する。
2. CREATE TABLE 文を使って表を定義する。
3. `.import` 文

CSV データをインポートする場合の `.import` 文は以下のように記述する。

```
sqlite> .separator ,
sqlite> .import data_for_import.csv table_name
```

一文目は属性値間の分離記号が ',' (カンマ) であることを指定する。SQLite はデフォルトでは '|' が分離記号として設定されているため、これを変更する。TSV ならこれを '\t' に変更すればよい。二文目でインポートするファイルとインポート先の表名をスペース区切りで記述する。SQLite のインポート方法では、入力ファイルのヘッダを除去する方法がないため、予めヘッダを削除するか、インポート後に DELETE 文を使ってヘッダに相当する行を削除するとよい。

<sup>45</sup>SQLite は関係データベースとして完全な機能を提供しているわけではないが、基本的な機能を有しており WEB アプリケーションで用いるデータサイズに向いているためよく使われている。より本格的な関係データベースとしては、PostgreSQL や MariaDB (MySQL のコミュニティバージョン) が無償で利用できる。有償のものでは Oracle がよく利用されている。

<sup>46</sup>どちらの方法が良いかは、データの量に依存するため一概にいいないことに留意されたい。

## 6 課題内容とレポート作成について

実験では、各班ごとに DMZ セグメントを備えたサブネットを構築し、Apache WWW サーバ上で SQL サーバと連携した WEB アプリケーションを動作させる。各班 3 台の計算機を使用する。そのうち 2 台（ルータと WWW サーバ）は、CentOS 8 の最小構成でインストール済みの状態から開始し、システム設定が必要となる。また、1 台（各班用 PC）は CentOS 8 の最小構成でインストール済みであり、システム設定は基本的に不要である。

ルータと WWW サーバは、デスクトップ環境がないため、コンソールから CLI (Command Line Interface) のコマンド入力と `vi` (`vim-minimal`) や `nano` 等のテキストエディタによる設定ファイルの編集により、設定作業を行う。コマンドの詳細な利用方法は、`man` コマンドで確認すること（操作方法は 4.9 節参照）。WEB の検索結果は、異なったバージョン等に関する場合も多いため、最終的には `man` の説明に従うこと。

尚、実験の都合上、SELinux を停止し、`yum` によるシステムアップデートは行わないものとするが、これらは一般には推奨されることではない。

- (1) 課題 1～課題 5 のレポートを作成するために、システムの出力は `script` コマンド等の手段を用いて、逐一ファイルに保存することが必要である。利用方法、制御文字の除去等は 4.9 節を参照すること。また、reboot コマンド等で再起動する前に、まず `exit` コマンドで `script` を終了すること。
- (2) 実験時には、`scp` コマンドによるネットワーク経由でのファイル転送が必須となる。利用方法は 4.5.1 節を参照すること。ただし、課題 2 終了後の段階では、ICE 端末と各班ルータ間でのみファイル転送が可能であり、課題 4 終了後に、WWW サーバ、各班用 PC から ICE 端末へ向けての転送も可能となる。ICE 端末側からローカルネットワークや DMZ ネットワークへ向けての接続は不可能である。また、各機器で利用可能なユーザ名が異なること、ホスト名が登録されていない機器には IP アドレスを利用する必要があることに注意する。
- (3) DNS サーバは、ICE の DNS サーバ (10.10.1.2, DNS 検索ドメイン `ice.nuie.nagoya-u.ac.jp`) を利用する (2.3.5 節参照)。

### 6.1 実験スケジュール

実験は 4 週にわたって行う。レポートを作成する期間を考慮すると、理想的には、次のようなスケジュールで実験を進めるのが望ましい。

第 1 週 [課題 1] 初期環境の確認

[課題 2] ネットワーク設定

[課題 3] DHCP サービスの設定

第 2 週 [課題 4] ファイアウォールの設定

[課題 5] WWW サービスの設定

第 3 週 [課題 6] SQL サーバと連携した Python WSGI アプリケーションの作成

第 4 週 [課題 6] SQL サーバと連携した Python WSGI アプリケーションの作成（続き）

### 6.2 実験課題

#### 【実験環境】

図 33 に実験で使用する機器とネットワーク構成を示す。

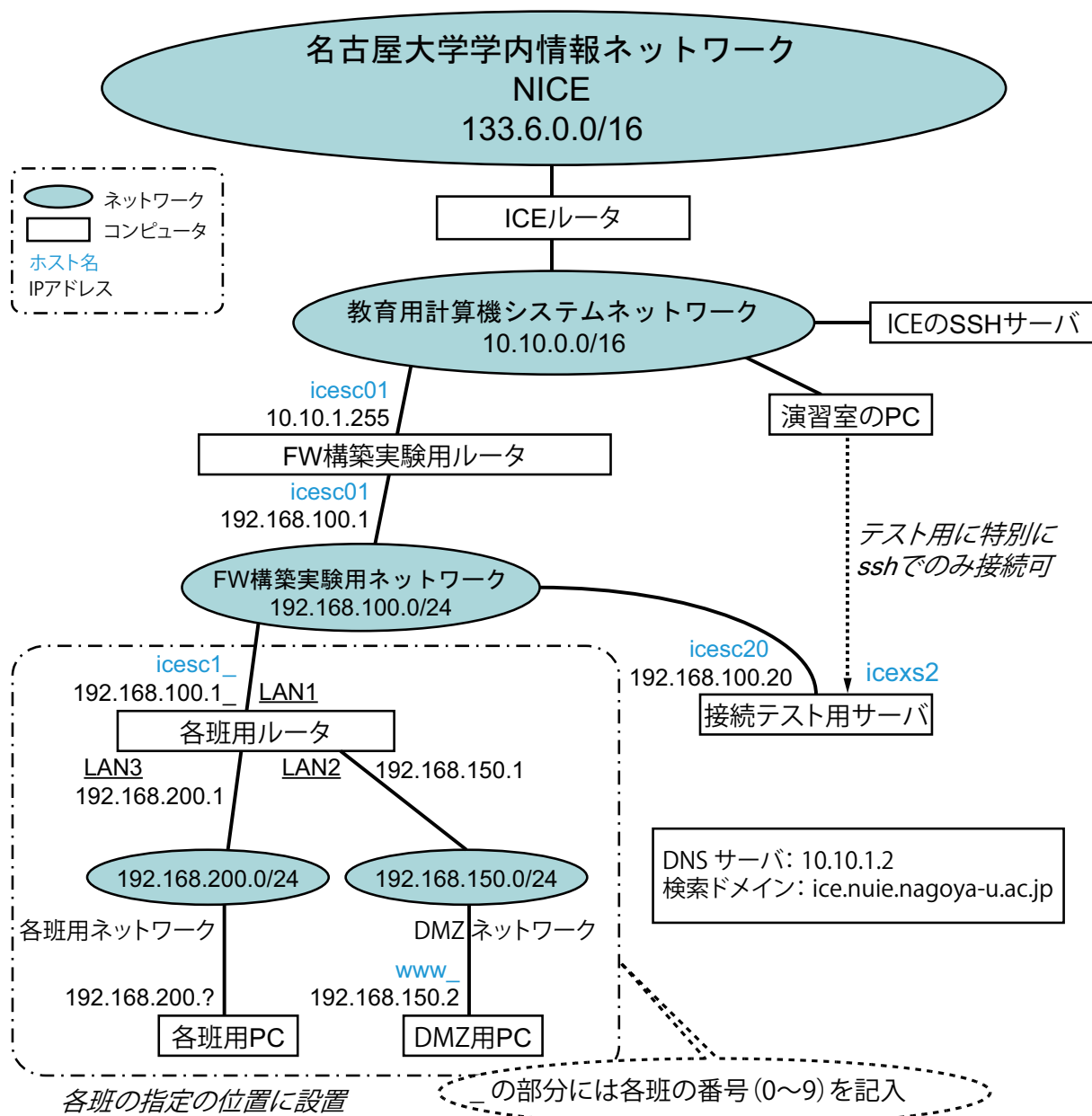


図 33: 実験用ネットワーク構成

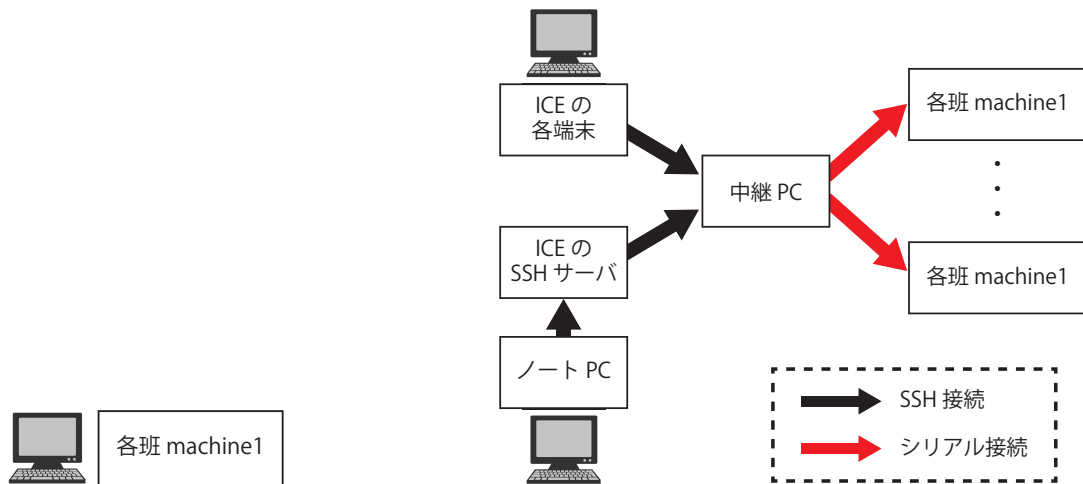


図 34: 通常のコンソール入出力

図 35: 実験のシリアル接続入出力

表 17: 各機器間の接続

利用可能な時期	可能な接続
常時	machine1 へのシリアル接続
課題 2 設定終了後	ICE ⇔ machine1 間の ssh/scp 接続
課題 3 設定終了後	machine1 ⇔ machine2, machine1 ⇔ machine3 間の ssh/scp 接続
課題 4 設定終了後	firewall で許可された ssh/scp 接続

各班には、図中の点線部分のネットワークがそれぞれ**サーバ室**に用意されており、2つのネットワーク（DMZ ネットワークと各班用ネットワーク）が各班用のルータ（以後 **machine1**）を介して接続されている。また、DMZ ネットワークには WWW サーバ（以後 **machine2**）が、各班用ローカルネットワークには各班用 PC（以後 **machine3**）が接続されている。これらの機器は単に物理的に接続されているのみであり、何の設定も行われていない状態では、通信を行うことはできない。

本実験では、コンソールから CLI (Command Line Interface) のコマンド入力と vi (vim-minimal) や nano 等のテキストエディタによる設定ファイルの編集により、図 33 のネットワークを実現する。図中の数値は設定すべき値を示しているため、参照しつつ課題を行うことが必要である。

実験の課題 2 まで、ネットワークインターフェイスが未設定な状態においては、ネットワーク経由の通信が不可能であることから、物理的に接続されているキーボード・モニタを用いた作業（図 34）が必要であるが、実はこれと等価な手段がシリアル接続経由での通信である。今回は各自のノート PC からの実験をも可能とするために、図 35 の接続を用意した。

図 35 中のシリアル通信より左側が machine1 に接続されているキーボード・モニタと等価である。以下の説明中で述べる「コンソールからの入力」とは図 35 の経路や ssh 接続による遠隔コマンド入力を指している。これによって、ICE の各端末やノート PC から完全に同様に実験が可能であるが、シリアル通信はキーボード・モニタを置き換えるものであり、一度に一名しか利用できないことに注意すること。接続方法の詳細は TACT に資料が準備されている。

表 17 に各機器間の接続とそれが可能となる時期を示す。実験の各段階において、可能な接続経路は異なっている。例えば、課題 3 以降では machine1 でコマンド入力する際に、シリアル接続経路でも ssh 接続経路でも同様に可能であり、ssh 接続経路の場合には複数人の同時接続も可能となる。ただし、オンラインで利用していると実感が薄れるかもしれないが、実際の機器は各班に 1 セットしか存在しない。各自が**入力するコマンドや保存するファイルは同一の機器を対象としている**のであるから、班員間の協調が必要であることを強調しておく。実験中は zoom のブレイクアウトルームを利用し、シリアル接続を行う者の端末を画面共有して、班内の情報共有を緊密にすることを想定している。



## 【課題 1】初期環境の確認（4.2, 4.5.3 節参照）

コンソールより, machine1 で以下の設定作業を行う.

### 1. Linux カーネルリリース番号の確認

通常, 同一ソフトウェアパッケージの複数のバージョンはインストールされないが, Linux カーネルパッケージは例外であり, 複数のカーネルが存在しうる. 現在動作している Linux カーネルを確認したければ, 以下のように入力する.

```
uname -r
```

### 2. ファイルシステムの確認

(a) ファイルシステムのマウントポイントを記述した設定ファイル `/etc/fstab` の内容を確認

```
cat /etc/fstab
```

(b) ディスクパーティションとマウントされているファイルシステムを確認

```
fdisk -l  
df -h
```

(c) 論理ボリュームの内容を確認

```
lvdisplay
```

### 3. ホスト名の設定

machine1 において, ホスト名の設定作業を行う.

ホスト名は, `icesc1_ice.nuie.nagoya-u.ac.jp` とする (ただし `_` は班番号 0~9).

(a) `hostname` コマンドでホスト名を設定し, 設定結果を確認

(b) ホスト名の恒久的変更のために, 設定ファイル `/etc/hostname` の内容を `vi` エディタで変更

(c) `exit` コマンドで `script` の終了後, `reboot` コマンドにより機器の再起動を行う.

これ以降は, **シェルのプロンプトにホスト名が表示される**. コマンド入力の際には, 正しい機器におけるコマンド入力であることを意識すること. 尚, machine2 にはホスト名 `www` が, machine3 にはホスト名 `internal` が既に設定されているため, プロンプトに表示される筈である.

### 4. SELinux の状態確認

SELinux は Linux のセキュリティ強化のために重要な機能であるが, 実験時のトラブルの元と成りうるため, 実験中は無効化するものとする. 尚, 無効化の作業は誤操作の危険を伴うため, 実験機器には既に無効化設定がなされており, 状態確認のみ行う.

(a) 現在の設定状況の確認

```
cat /etc/sysconfig/selinux  
getenforce
```



## 【課題 2】ネットワーク設定 (2.1, 2.4, 4.4.1 節参照)

この課題では machine1 において、nmcli コマンドを利用して、ネットワークインターフェイスの接続設定を行う。尚、IPv6 は利用しないため設定不要である。

### 1. 現在の状況の確認

以下では、デバイス device と 接続 connection の相違に注意すること。

#### (a) ネットワークデバイスの確認

machine1 LAN1~LAN3 のデバイス名を確認する。デバイスノードは 4.2.1 節で述べたように自動的に生成されるが、machine1 では LAN1~LAN3 の番号はデバイス名の番号と一致している。lo は loopback network interface を意味し、ホスト自身に戻る仮想的なデバイスであり設定は不要である。

課題 3 の終了後に行う machine2, machine3 のネットワーク情報確認においては、wifi デバイスも確認できるが、本実験では使用しない。

```
nmcli device status
nmcli device show
```

#### (b) ネットワーク接続の確認

```
nmcli connection show
```

上で確認したデバイス名と同一の接続名が作成されていることを確認する（下記仕様の 2）。接続（設定）は存在するが、まだ未接続の状態であることが確認できる。

### 2. 接続設定

以下の仕様を満たすように、nmcli コマンドを利用して、接続設定の修正を行う。4.4.1 節で述べたように、サブコマンド help の利用で利用例が表示されるため参考にせよ。

また、nmcli con show 接続名 により、設定可能な内容と現在の設定値が表示されるが、このうち connection.xxx と ipv4.xxx の一部が設定すべき項目である。これらのオプションの正確な内容は、man nm-settings で確認することができる。

- (1) ネットワークインターフェイス設定時におけるゲートウェイとは、当該インターフェイスが含まれるネットワークからの出口、すなわちルータの IP アドレス（ネクストホップ）である（2.4 節参照）。自分自身がルータである場合には外部ネットワークに至るインターフェイスでのみゲートウェイを指定すれば良いことに注意すること。
- (2) nmcli コマンドで設定変更後に変更が反映されない場合は、接続の停止・開始  
nmcli con down 接続名, nmcli con up 接続名 を行うこと。
- (3) DNS を間違えて設定した後、設定仕直しても /etc/resolv.conf に変更が反映されない場合は、NetworkManager の再起動 systemctl restart NetworkManager を行うこと。
- (4) タイミングによっては、NetworkManager によって、接続「有線接続 1」等が自動生成され、さらに文字化けにより “■■■■ 1” 等の表示が確認されることがある。この場合には、nmcli con show で確認した 32 桁の UUID を指定して、nmcli con del <UUID> で削除後、  
nmcli con add type ethernet con-name 接続名 ifname デバイス名  
により、新規接続を追加する。

#### machine1 ネットワーク接続仕様

- インターフェイス LAN1 を外部ネットワーク用, LAN2 を DMZ ネットワーク用, LAN3 を内部ネットワーク用として利用
- 接続名はデバイス名と同一の Ethernet 接続
- ipv4 固定アドレス, ネットマスクを手動で割り当て
- DNS サーバを設定 (LAN1 のみ)
- デフォルトゲートウェイを適切に設定 (LAN1 のみ)
- OS 起動時の自動接続を指定

設定手順は以下の様になる.

- (a) machine1 の LAN1 に対して, `nmcli con show` 接続名 により, 現在の設定値を確認
- (b) machine1 の LAN1 に対して, `nmcli con mod` コマンドにより,
  - `ipv4.method` と `ipv4.addresses` の設定を変更 (仕様の 3 番目に対応, 同時変更が必要)
  - `ipv4.dns` と `ipv4.dns-search` の設定を変更 (仕様の 4 番目に対応)
  - `ipv4.gateway` の設定を変更 (仕様の 5 番目に対応)
  - `connection.autoconnect` の設定を変更 (仕様の 6 番目に対応)

`nmcli con mod help` コマンドの結果を参考にする.

- (c) machine1 の LAN2, LAN3 に対して, `nmcli con mod` コマンドにより, `ipv4.method`, `ipv4.addresses`, `connection.autoconnect` の設定を変更
- (d) machine1 において, `nmcli con show` 接続名 コマンドにより, 設定内容を確認
- (e) `nmcli dev status`, `nmcli con show` コマンドにより, すべての接続を確認

### 3. 各種情報の確認

接続が正しく行われているかを以下のコマンドで確認する.

<code>ip addr show</code>	(各デバイスの設定の確認)
<code>ip route</code>	(ルーティング情報の確認)
<code>cat /etc/resolv.conf</code>	(DNS 設定の確認)

#### 調査課題

- (1) TCP パケットのヘッダ情報, IP パケットのヘッダ情報にはどのような情報が格納されているかを調査せよ.
- (2) TCP/IP 通信におけるブロードキャストの役割について調査せよ. ブロードキャストには, どのような特徴があるかを述べよ.
- (3) 各班ローカルネットワークの PC1 (192.168.200.2) から, C クラスサブネット ICE の PC2 (10.10.1.20) に `ssh` コマンド `ssh 10.10.1.20` で接続を行った. この時, TCP パケットが PC2 に到達するまでの過程において, 必要な情報がどのような手段で取得されるかを調査課題 1, 課題 2, 課題 4 の結果を参照しつつ, 詳細に述べよ. ルーティング情報とブロードキャストの役割については, 必ず触れること.

### 【課題 3】DHCP サービスの設定・起動 (2.3.6, 4.3.4 節参照)

この課題では machine1 で DHCP サーバを動作させ, DMZ ネットワークと各班用ネットワークの両方のサブネット接続機器への設定情報の提供を行う.

共通パラメータ

```
subnet network_address netmask netmask {  
    サブネット固有のパラメータ  
}  
  
host hostname {  
    ホスト固有のパラメータ  
}
```

図 36: dhcpd.conf の書式

以下の設定を machine1 で行い、サービス起動後は、machine2, machine3 で確認作業を行う。

dhcpd.conf の設定については `man dhcpd.conf` を参照すること。Red Hat Enterprise Linux 6 導入ガイド [http://access.redhat.com/documentation/ja-jp/red\\_hat\\_enterprise\\_linux/6/html/deployment\\_guide](http://access.redhat.com/documentation/ja-jp/red_hat_enterprise_linux/6/html/deployment_guide) の 16.2.1 節の設定例も参考になる（サイドバーで Expand all 後に 16.2.1 節を参照）。

#### 1. firewall の停止

サービスの設定・確認作業を円滑に進めるため、一時的に firewall を停止する（一般には推奨できない）。status が inactive と成っていることを確認する。尚、status の確認は q キーを押せば終了する。

```
systemctl stop firewalld  
systemctl status firewalld
```

#### 2. DHCP サーバのインストールと設定

##### (a) DHCP 関連パッケージの確認と DHCP サーバのインストール

最初のインストール時には、GPG (Pretty Good Privacy 互換実装) キーのインポート許可を求められるので、許可すること。

```
yum list dhcp-*  
yum info dhcp-server  
yum install dhcp-server
```

##### (b) 設定ファイルの修正

以下の仕様を満たすように DHCP サーバの設定ファイル `/etc/dhcp/dhcpd.conf` を適切に設定する。dhcpd.conf の設定書式は、(1) 共通パラメータ設定部分（man の設定例では、global parameters）、(2) サブネット設定部分、(3) ホスト設定部分の 3 種類から構成されている（図 36）。

仕様を満たすように `range`, `option routers`, `option domain-name-servers`, `option domain-name`, `default-lease-time`, `max-lease-time`, `hardware ethernet`, `fixed-address` 等のパラメータを適切に設定する必要がある。パラメータの詳細については、`man dhcpd.conf` を参照すること。

### DHCP サーバの仕様

- DMZ ネットワーク, 及び, 各班用ネットワークに IP アドレスをリース
- リース IP アドレスは DMZ ネットワークが 192.168.150.100~192.168.150.250, 各班用ネットワークが 192.168.200.100~192.168.200.250
- ゲートウェイ, ネットマスク, DNS サーバを指定
- リース期間はデフォルト 1 時間, 最大 1 日
- machine2 は MAC アドレス指定で固定 IP アドレスを割り当て  
また, リース期間は 30 日

- (c) DHCP サーバ (サービス名 `dhcpcd`) を起動 (4.3.4 節の表 8 を参照)
- (d) DHCP サーバの状態が `active` (動作中) であることを確認 (4.3.4 節の表 8 を参照)
- (e) DHCP サーバのブート時自動起動を指定 (4.3.4 節の表 8 参照)
- (f) machine2, machine3 (192.168.200.100) の生存を `ping` コマンドで確認
- (g) machine2, machine3 に `ssh` でログインし, ネットワーク設定が行われていることを確認

<code>ip addr show</code>	(各デバイスの設定の確認)
<code>ip route</code>	(ルーティング情報の確認)
<code>cat /etc/resolv.conf</code>	(DNS 設定の確認)

- (h) 結果提出

`dhcpcd.conf` ファイルを結果提出用ディレクトリに提出する.

提出先は `/pub1/jikken/cs-net/students/group__` (\_\_ は班番号 0a ~ 9a, 0b ~ 9b) .

- (1) 本課題における設定は DHCP クライアントに対するネットワーク設定である. 前課題でのゲートウェイと混同しないように注意せよ.
- (2) DHCP サーバ起動後に `dhcpcd.conf` の設定内容を変更した場合, `systemctl restart dhcpcd` でサーバを再起動しない限り設定は反映されないの, 注意すること.
- (3) DHCP サーバを誤設定して起動すると, machine2, machine3 側は誤設定の情報をリースされ, DHCP サーバ設定を修正して再起動 (`restart`) してもリース期間中は反映されないことがある. このような場合には, `nmcli` コマンドによる接続の停止・開始 (課題 2 参照), または machine1 の再起動を行うこと.

### 一週目終了時の確認項目

課題 3 まで終了したら, machine1 で `exit`, `reboot` を行う. その後, TACT の確認用資料と以下の実行結果を比較し, 一致することを確認後に実験を終了する. `script` コマンドで保存したログファイルの内容を確認すること.

1. machine1, machine2, machine3 において, `nmcli con show` を実行した結果
2. machine1, machine2, machine3 において, `ip addr`, `ip route` を実行した結果
3. machine1, machine2, machine3 において, `cat /etc/resolv.conf` を実行した結果

表 18: ポート番号とその用途

ポート番号	名前と目的
0～1023	ウェルノウンポート (だいたいプロトコルが決められている)
1024～49151	登録ポート (アプリによって割り当てられている)
49152～65535	ダイナミック／プライベートポート (自由に使える)

No.	Protocol	用途
20	ftp-data	ファイル転送 (データ本体)
21	ftp	ファイル転送 (コントロール)
22	ssh	シェル: SSH (セキュア)
25	smtp	メール送配信: SMTP
53	domain	DNS
67	bootps	DHCP サーバ
68	bootpc	DHCP クライアント
80	http	WWW
110	pop3	メール受信 (POP)
143	imap	メール受信 (IMAP)
443	https	WWW (セキュア)

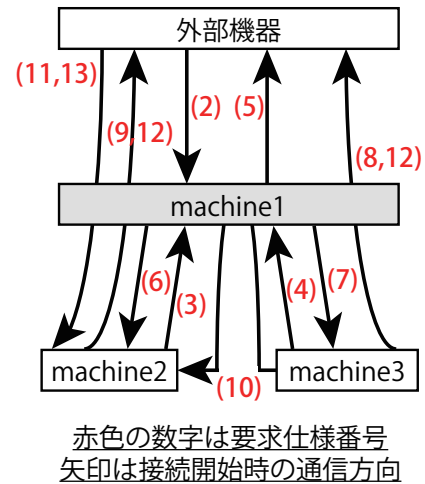


図 37: machine1 ファイアウォール仕様

#### 【課題 4】ファイアウォールの設定 (3 章, 4.4.3 節参照)

この課題では machine1, machine2 のファイアウォールを設定し, 内部・外部端末から接続の確認を行う。

3 章で述べたように, iptables はポート番号等の情報を利用してパケットフィルタリングを行う。ポートにはユーザが自由に使えるものと, すでに決まったサービスのために割り当てられているものの 2 種類がある (表 18 参照)。また, ポートがどのように使われているかは, /etc/services で確認できる。

##### 1. machine1 のファイアウォール設定準備

machine1 にファイアウォールの設定を行う前に, 以下の準備作業を行う。

###### (a) ネットワーク接続設定の確認

先週の接続設定が問題ないことを, machine1, machine2, machine3 で以下のコマンドにより確認する。接続設定が間違っていると今後の作業が行えなくなるため, しっかり確認する。

```
ip addr show      (各デバイスの設定の確認)
ip route          (ルーティング情報の確認)
cat /etc/resolv.conf (DNS 設定の確認)
```

###### (b) パケット転送の有効化

設定ファイル /etc/sysctl.d/10-ipv4.conf をエディタで作成, 以下を追加し,

```
net.ipv4.ip_forward=1
```

以下のコマンドで有効化する。

```
sysctl --system
reboot
```

(c) firewalld から iptables-services への変更

ルータでは要求仕様が複雑になるため、詳細なルールを記述可能な iptables ルールの直接記述を行う。このために、以下のコマンドで Netfilter のフロントエンドを firewalld から iptables に変更する。

```
systemctl stop firewalld
systemctl disable firewalld
yum install iptables-services
systemctl start iptables
systemctl status iptables
systemctl enable iptables
```

(d) サンプルの実行

ICE のホスト ssh のディレクトリ /pub1/jikken/cs-net に用意した iptables 設定サンプルファイル iptables-sample.sh を scp (4.5.1 参照) で machine1 にコピーし、実行する。iptables-sample.sh の内容は、指導書サンプルリスト A.1 に記載されているので参照すること。サンプルリスト中の \ 記号は改行文字のエスケープ、すなわち次の行への継続を意味する。

尚、iptables-sample.sh は iptables 設定初期化のためのスクリプトとして重要なため、消さずに保存しておき、問題発生時に適宜実行すること。

```
scp ユーザ名@ssh.ice.nuie.nagoya-u.ac.jp:/pub1/jikken/cs-net/iptables-sample.sh ~
```

iptables スクリプトは sh コマンドで実行することにより設定が行われる。

```
sh iptables-sample.sh
```

(e) iptables の設定状況を確認

```
iptables -L -n
iptables-save      (現在設定されているルールを表示)
```

## 2. machine1 のファイアウォール設定

machine1 の要求仕様は以下の通りである。図 37 も参照すること。尚、サンプルファイル iptables-sample.sh には要求仕様の一部を既に実装済みである。

#### machine1 のファイアウォール要求仕様

- (1) INPUT/OUTPUT/FORWARD チェーンの初期設定をすべて DROP にする。
- (2) 外部ネットワークからルータへの接続は、SSH のみ許可し、その他の接続はすべて拒否する。
- (3) DMZ ネットワークからルータへの接続は、DHCP, ping のみ許可する。
- (4) ローカルネットワークからルータへの接続は、SSH, DHCP, ping のみ許可する。
- (5) ルータから外部ネットワークへの接続は、SSH, HTTP/HTTPS, DNS, ping のみ許可する。
- (6) ルータから DMZ ネットワークへの接続は、SSH, ping のみ許可する。
- (7) ルータからローカルネットワークへの接続は、SSH, ping のみ許可する。
- (8) ローカルネットワークから外部ネットワークへの接続は、SSH, HTTP/HTTPS, DNS, ping のみ許可する。
- (9) DMZ ネットワークから外部ネットワークへの接続は、SSH, HTTP/HTTPS, DNS, ping のみ許可する。
- (10) ローカルネットワークから DMZ ネットワークへの接続は、すべて許可する。
- (11) 外部ネットワークから DMZ ネットワークへの接続は、HTTP/HTTPS のみ許可する。
- (12) ローカルネットワーク、DMZ ネットワークから外部ネットワークへの接続は SNAT を行う。
- (13) machine2 の HTTP/HTTPS サーバをルータの IP アドレスで公開する (port forwarding) DNAT の設定を行う。

- (1) “接続” とは、クライアントからサーバに向けて開始され、それに引き続くすべての通信を指し、サーバからクライアントへの返答パケットも含まれる。すなわち **通信は双方向的である**。通常、クライアントからサーバへのパケットとサーバからクライアントへのパケットは条件が異なるため、それぞれのルールを考える必要がある。

上記仕様を満たすように、以下の手順により、iptables の設定ファイルを作成する。iptables-sample.sh をコピーし、ターゲット ACCEPT のルール追加、実行、確認を繰り返し、設定ファイルを完成させる。確認を十分に行っていれば、ルールの追加により前の確認が無効になることはない。すなわち、設定の後戻りは発生しない筈である。

ICE の端末でファイルを修正し、scp コマンドで machine1 の root のホームディレクトリ (/root) にコピーする手順が効率的である。machine1 では root のホームディレクトリでスクリプトを実行すれば良い。

ICE の端末から各班の machine1 にファイルをコピーするためには、以下を実行する。

```
scp ファイル名 root@icesc1_:
```

- (a) ルータからの/ルータへの接続に関するサンプルリストの確認

仕様 1 ～ 7 はすべて実装済みである。仕様 1 及び、2 ～ 7 に対応するルータからの接続とルータへの接続をサンプルリスト中の INPUT, OUTPUT チェーン設定で確認する。SSH/HTTP(S)/DNS/DHCP サービスに関しては、行きの通信が 2 ルール、戻りの通信が 1 ルールで記述されており、ping に関しては、行き/戻りの通信が各 1 ルールで記述されている。ただし、複数の接続に関するルールが同一のルールとなる場合がある。リスト中のどのルールの組がどのサービスの行き/戻りの通信と対応しているかを良く確認しておくこと。

- (b) 接続確認



仕様 2 ～ 7 に対応する接続が可能であることを nslookup/dig/host, ssh, lynx, ping コマンドによって実際に確認する。外部ネットワークのサーバには、ICE の ssh サーバと WWW サーバ、情報学研究科の WWW サーバを利用すること。DNS が利用できない場合、他のサービスでもホスト名による接続は一切できないため、最初に DNS の確認を行うこと。DNS サーバの設定は課題 2, 3 で行っているため、これを間違えていた場合には確認に失敗する事に注意する。ssh はユーザ名に注意する。

確認すべき内容は、以下のとおりである。

- 外部ネットワークから machine1 への接続 (SSH)
  - machine2 から machine1 への接続 (ping)
  - machine3 から machine1 への接続 (SSH, ping)
  - machine1 から外部ネットワークへの接続 (DNS, SSH, HTTP/HTTPS, ping)
  - machine1 から machine2 への接続 (SSH)
  - machine1 から machine3 への接続 (SSH, ping)
- (c) ローカルネットワークから外部/DMZ ネットワークへの接続に関するサンプルリストの確認  
仕様 8 と仕様 12 の前半部分は実装済みである。仕様 8 に対応するローカルネットワークから外部ネットワークへの接続をサンプルリスト中の FORWARD チェーン設定で確認する。また、仕様 10 は実装済みである。仕様 10 に対応するローカルネットワークから DMZ ネットワークへの接続をサンプルリスト中の FORWARD チェーン設定で確認する。行きの接続が 1 ルール、戻りの接続が 1 ルールで記述されている。ただし、複数の接続に関するルールが同一のルールとなる場合がある。
- (d) 接続確認  
仕様 8, 12 前半、仕様 10 に対応する接続が可能であることを実際に確認する。  
確認すべき内容は、以下のとおりである。
- machine3 から外部ネットワークへの接続 (DNS, SSH, HTTP/HTTPS, ping)
  - machine3 から machine2 への接続 (SSH, ping)
- (e) DMZ ネットワークから外部ネットワークへの接続に関する設定の追加  
仕様 9, 12 後半部分の iptables ルールを追加する。仕様 8 の実装を参考にせよ。
- (f) 接続確認  
仕様 9, 12 後半に対応する接続が可能であることを実際に確認する。  
確認すべき内容は、以下のとおりである。
- machine2 から外部ネットワークへの接続 (DNS, SSH, HTTP/HTTPS, ping)
- (g) 外部ネットワークから DMZ ネットワークへの接続に関する設定の追加  
仕様 11, 13 の iptables ルールを追加する。DNAT については、3.6.2 節を参照すること。尚、接続確認は課題 5 で行う。
- (h) ファイアウォール動作状況の最終確認  
ファイアウォールの動作が仕様のとおりにあることを、内部・外部機器から接続することで確認する。ただし、machine2 の WWW サーバ公開確認については、課題 5 の終了後に行う。  
確認すべき内容は、以下のとおりである。
- machine1 から外部ネットワークへの接続 (SSH, HTTP/HTTPS, DNS, ping)
  - machine2 から外部ネットワークへの接続 (SSH, HTTP/HTTPS, DNS, ping)
  - machine3 から外部ネットワークへの接続 (SSH, HTTP/HTTPS, DNS, ping)
  - machine3 から machine1 への接続 (SSH, ping)
  - machine3 から machine2 への接続 (SSH, ping)
  - machine2 から machine3 への接続 (SSH 拒否, HTTP/HTTPS 拒否, ping 拒否。パケット DROP のため無反応となる)
  - 外部ネットワークから machine1 への接続 (SSH)

- (i) 最終状態を保存

```
iptables-save > /etc/sysconfig/iptables
```

- (j) 結果の提出

iptables-sample.sh の最終修正ファイルを結果提出用ディレクトリに提出する.

提出先は /pub1/jikken/cs-net/students/group\_\_ (\_\_ は班番号 0a ~ 9a, 0b ~ 9b) .

### 3. machine2 のファイアウォール設定

#### machine2 のファイアウォール要求仕様

- SSH, HTTP/HTTPS, DHCPv6client, すべての ICMP パケットのみ許可し, その他の接続はすべて拒否する

machine2 では, ファイアウォールのルールが簡単なため, firewalld サービスを利用し, デフォルトの public ゾーンを元に設定の修正を行う.

- (a) firewalld の動作状態を確認

```
systemctl status firewalld
```

- (b) 現在のゾーンの許可状態を確認

```
firewall-cmd --list-all
```

- (c) 必要なサービス許可設定の追加

```
firewall-cmd --add-service=dhcpv6-client
firewall-cmd --add-service=http
firewall-cmd --add-service=https
firewall-cmd --list-all
```

- (d) 各設定の永続化

```
firewall-cmd --permanent --add-service=dhcpv6-client
firewall-cmd --permanent --add-service=http
firewall-cmd --permanent --add-service=https
```

- (e) firewalld サービスの再起動

```
systemctl restart firewalld
```

#### 調査課題

- (4) DNS (Domain Name System) の詳細に関して, 最近のセキュリティ強化の動向も含めて, 調査せよ.

### 【課題 5】WWW サービスの設定・起動 (4.5.2 節参照)

この課題では machine2 で WWW サーバを動作させる.

以下の設定を machine2 で行い, サービス起動後は, machine3 と ICE の端末で確認作業を行う.

1. firewalld の停止

サービスの設定・確認作業を円滑に進めるため, 一時的に firewalld を停止する.

## 2. WWW サーバのインストール

Apache WWW サーバと https 対応モジュールをインストールする。

```
yum info httpd
yum info mod_ssl
yum install httpd
yum install mod_ssl
```

## 3. SSL/TLS 自己署名証明書の作成

SSL/TLS 通信を行うために、openssl コマンドにより自己署名証明書を作成する。

**/etc/pki/tls/certs に移動し**、以下のコマンドを入力する。

### (a) 秘密鍵の作成

```
openssl genrsa -aes128 2048 > server.key
```

パスフレーズは適当な文字列を入力する。

### (b) パスフレーズの除去

秘密鍵を AES で暗号化している場合、WWW サーバの起動時にパスフレーズの入力を要求される。WWW サーバ自動起動のためには、これは不都合であるため、セキュリティ性能は劣化するが、往々にして以下のようにパスフレーズ除去が行われる。

```
openssl rsa -in server.key -out server.key
```

### (c) CSR の作成

```
openssl req -utf8 -new -key server.key -out server.csr
```

### (d) 有効期間 1 年のサーバ証明書の作成

```
openssl x509 -in server.csr -out server.crt -days 365 -req -signkey server.key
```

## 4. WWW サーバの設定ファイルにサーバ証明書を指定

設定ファイル `/etc/httpd/conf.d/ssl.conf` の一部のディレクティブの値を以下のように変更する（**該当ディレクティブ以外の値は変更しないように注意**）。

```
SSLProtocol +TLSv1.2
SSLCertificateFile /etc/pki/tls/certs/server.crt
SSLCertificateKeyFile /etc/pki/tls/certs/server.key
```

## 5. WWW サーバの主設定ファイルを修正

Apache WWW サーバ主設定ファイル `/etc/httpd/conf/httpd.conf` の `ServerName` ディレクティブを各班 machine1 の外部 LAN インターフェイスの FQDN に設定する。

## 6. サンプル WEB ページのコピー

ICE SSH サーバの `/pub1/jikken/cs-net/` 以下にある `index.html` と `default.css` を machine2 の WWW サーバドキュメントルート `/var/www/html` に `scp` コマンドを用いてコピーする。

## 7. WWW サーバ（サービス名 httpd）の起動

```
systemctl start httpd
```

起動時にエラーが発生した場合は、サービスの `status` を確認し、エラー箇所を特定する。

8. machine3 と ICE (演習室端末か icexs2 のいずれか) から http, https 両方の動作を確認  
ICE からは machine1 のホスト名を, machine3 からは machine2 の IP アドレスを URL に用いる. https は自己署名証明書のためブラウザでエラーとなるが, lynx ではそのまま続けることで接続される. 日本語のテストページが表示される筈である. GUI ブラウザ Firefox の場合は「詳細情報」から「危険性を承知で続行」することで接続可能となるため, 接続後に証明書の詳細を確認すること.
9. WWW サーバのブート時自動起動を指定 (4.3.4 節の表 8 参照)

- (1) WWW サーバの動作確認時に接続ができない場合は, 原因が課題 5 の httpd 設定に基づくものか, 課題 4 仕様 11, 13 に基づくものかを特定し対応する. machine3 からの接続に問題がない場合には, 課題 4 の設定ミスが原因である.
- (2) 課題 4 の iptables 設定を修正し直した場合には, iptables-save の実行と最終設定ファイルの再提出を忘れないこと.

#### 調査課題

- (5) SSL/TLS を利用して通信路の暗号化を行う HTTPS 以外のプロトコルについて調査せよ.

### 【課題 6】SQL データベースと連携した Python WSGI WEB アプリケーションの作成

WEB サーバ上で SQL データベースと協調して動作する WEB アプリケーションを Python 言語で作成する.

#### 1. Sqlite3 での SQL 実行

本課題で作成する WEB アプリケーションでは SQL データベースを利用してデータの管理を行う. 最終的に Python プログラムから SQL データベース sqlite3 を利用することを想定する. SQL や sqlite3 についての説明は 5.3 節を参照されたい. ここではまず sqlite3 の基本的な使い方を確認するために, コマンドラインから対話モードで使ってみる. 各自の端末で以下の簡単な実行例をトレースせよ.

データベース名を kadai6.db として sqlite3 を起動する.

```
sqlite3 kadai6.db
```

まず users という名前の表を作成する. users は name と age という属性をもたせる.

```
sqlite> create table users (name text, age integer);
```

作成した表に ('John', 20), ('Bob', 36) という 2 つの行を挿入する.

```
sqlite> insert into users values ('John', 20), ('Bob', 36);  
sqlite> select * from users;
```

age 属性の値が 30 より大きい行を削除する.

```
sqlite> delete from users where age > 30;  
sqlite> select * from users;
```

aozora という名前の別の表を作成する.

```
sqlite> create table aozora (bookID integer, title text, author text);
```

ICE の共有ディレクトリ/pub1/jikken/cs-net/以下にある aozora-min.csv という csv ファイルのデータ (本番号, 題目, 著者の情報) を aozora という表にインポートする. その後, 著者が'夏目'で始っている本のデータだけを選択し, その題目と著者だけを抽出してみる.

```
sqlite> .separator ,  
sqlite> .import /pub1/jikken/cs-net/aozora-min.csv aozora  
sqlite> select title, author from aozora where author like '夏目%';
```

終了するときは .quit か .exit を実行する.

```
sqlite> .quit
```

## 2. WSGI サンプルコードの動作確認

sqlite3 モジュールを利用した簡単な WSGI サンプルコードを実行してその動作を確認する. machine2 において以下の (a)-(d) の作業を行う.

- (a) python3 及び WSGI 対応の Apache モジュール (python3-mod\_wsgi) をインストールする.

```
yum install python36  
yum install python3-mod_wsgi
```

- (b) /etc/httpd/conf.d/wsgi.conf を作成して, 以下の 2 行を書き込む. その後, httpd サーバを再起動する.

```
WSGIScriptAliasMatch ^/app/([^/]+)/ /var/www/app/$1/  
Alias /static /var/www/html
```

以上の記述は, [http://icesc1\\_/app/test/xxx.wsgi](http://icesc1_/app/test/xxx.wsgi) などのリクエストに対して実行される WSGI スクリプトの場所<sup>47</sup>と, [http://icesc1\\_/static/xxx.css](http://icesc1_/static/xxx.css) などのリクエストに対して返すファイルの場所を指定している.

- (c) /var/www/app/test というディレクトリを作成して, その所有者を apache に変更する.

```
chown apache:apache /var/www/app/test
```

- (d) WSGI サンプルコード (/pub1/jikken/cs-net/sample.wsgi) を/var/www/app/test に, サンプル CSS ファイル (/pub1/jikken/cs-net/default.css) を/var/www/html 以下にコピーする.

以上の設定後, ICE のネットワーク上の端末のブラウザで [http://icesc1\\_/app/test/sample.wsgi](http://icesc1_/app/test/sample.wsgi) にアクセスすると, サンプルコードが作成する WEB ページが確認できる. サンプルコードは, 学生番号と氏名を入力するフォームを含む WEB ページを返し, 学生番号と氏名の組が送信されてきた場合にはデータベースにそれらを格納してそれまでに追加された組すべてを表示した WEB ページを返す. 5.2 節および 5.3.3 節を参考にしてサンプルコードの内容も確認せよ. サンプルコードは, 指導書サンプルリスト [A.2](#) に記載されている.

なお, 各自の端末でも実行確認できるように, サンプルを python コマンドで直接実行した場合にはリファレンス WEB サーバを起動するようにしてある. python の実行環境があれば Apache を起動しなくても動作確認することができる. 手順としては, 各自の端末に sample.wsgi と default.css をダウンロー

<sup>47</sup>詳細は <https://modwsgi.readthedocs.io/en/develop/user-guides/configuration-guidelines.html> を参照

ドして、sample.wsgi が置いてあるディレクトリに新たに static というディレクトリを作成してそこに default.css を置く。そして、WSGI サンプルを実行して (python3 sample.wsgi で実行)、それを実行した端末上のブラウザで http://localhost:8080/ にアクセスすることで、WEB ページが確認できる。

### 3. WEB アプリケーションの作成

SQL データベースと協調して動作する WEB アプリケーションを各自で作成する。作成するアプリケーションは以下に挙げる要件を最低限満たすようにすれば、その他は自由に作成してよい。

- 何らかのデータを管理するサービスを提供すること。WEB 上でのユーザによる操作に応じてデータの検索と更新が伴うものとし、SQL データベースを利用して実現すること。利用するデータは自由に用意して構わない。TACT のリソースにサンプルとして青空文庫の csv データを用意しているのでそれをあらかじめインポートして用いてもよい (aozora.csv, aozora-min.csv, aozora-header.txt)。

以上の要件に加えて、アプリケーションによって動的に生成される WEB ページの表示属性を CSS に分離することを推奨する。すなわち、HTML タグに ID 名/クラス名を付加し、CSS のみの変更でレイアウトを自由にコントロールできるようにする。レイアウトもなるべく見やすくなるように工夫することが望ましい。HTML および CSS については 5.1 節を参考にせよ。

**提供するサービスの一例** 文庫本の読書状況を管理するためのアプリケーションを例として挙げる。

- 新たな文庫本の情報 (タイトル, 著者, 出版社など) を入力して追加できる。
- タイトルや著者をキーとして文庫本の検索ができる。検索条件にマッチしたすべての本の著者名, 作品名, 出版社が表形式で表示される。
- 検索結果の表の各行にチェックボックスがあり、本の情報を選んで削除できる。

本課題として以上の例そのものあるいは機能追加をしたものを作成しても構わない。

#### 補足事項

- 本実験では Flask や Django などの Python WEB アプリケーションフレームワークは利用しないこと。

#### 調査課題

(6) SQL インジェクションなどの Web アプリケーションのセキュリティを脅かす代表的な脆弱性について調査せよ。

## 6.3 レポートについて

レポートは 2 回に分けて提出する。初回レポートは課題 1~5 と調査課題 1~5, 最終レポートは課題 6 と調査課題 6 についてである。各回の提出日ならびに注意事項については、学生実験の Web ページを確認すること。レポートは、TACT の課題ツールを通じて提出すること。グループ実験については、各自の果たした役割などを明確にすること。

レポートでは、各実験に対して、

- 実験の方法
- プログラムの処理手順, プログラム作成上の工夫
- 実験で得られた結果
- 結果に対する考察

をまとめること。なお、レポートの書き方については、実験レポートの作成に関する動画で説明された点に注意すること。

## 課題 1～5 のレポートに関する注意点

課題 1～5 のレポートについては、以下の点に注意して、レポートを作成すること。

1. 実験時に行った設定内容やコマンド入力、およびシステムの出力結果や確認結果を、正確にレポートに記述すること。尚、上記のコマンド入力やシステムの出力結果等には画像添付を認めないので、必ずテキストで記述すること。
2. 得られた結果のみではなく、何故そのような結果が得られたかを、指導書の記載事項や各自の調査結果に基づいて、分かりやすく記述すること。
3. 課題 5 の machine1 のファイアウォールの設定に関しては、設定に用いた iptables ルールと要求仕様との対応を取りつつ、利用したパラメータのレベルまで、その設定内容の意味を文章によって詳しく解説すること。
4. 調査課題に関しては、参考にした文献を必ず記載すること。また、各調査課題について、**最低 1 ページ**は記述すること。

## 課題 6 のレポートに関する注意点

課題 6 については、TACT の課題ツールで以下のものを提出すること。

- レポート（PDF ファイル）
  - － 作成したアプリケーションの概要と機能の説明
  - － サービスの利用方法（起動方法から、ブラウザに表示されたページ上の操作も含める）
  - － 工夫した点、未解決のバグ、課題
  - － 調査課題 (6)
- プログラムなどの ZIP アーカイブファイル

レポートにはプログラムのソースコードを掲載しなくてよい。ただし、レポートを書く上で必要があれば、部分的にプログラムを載せて説明を加えても構わない。



## 参考図書・URL

- [1] IETF と RFC: 社団法人日本ネットワークインフォメーションセンター (JPNIC),  
<http://www.nic.ad.jp/ja/tech/rfc-jp.html>
- [2] 上原政二, “標準 LAN 教科書・上”, アスキー出版.
- [3] 羽山博, “UNIX ネットワークプログラミング”, オーム社.
- [4] 山本和彦, “ハッピーネットワークング”, アスキー出版.
- [5] Douglas Comer, David Stevens, “TCP/IP によるネットワーク構築 Vol.1.2”, 共立出版.
- [6] 小俣光之, “C 言語による TCP/IP セキュリティプログラミング”, ピアソン・エデュケーション, 2002.
- [7] 高田美樹, “Java 完全マスターブック”, 技術評論社, 2004.
- [8] JPCERT/CC, <http://www.jpccert.or.jp/>
- [9] “情報セキュリティ入門”, 富士通オフィス機器株式会社, 2002.
- [10] 情報処理基礎講座 “セキュリティポリシー”, 電子開発学園出版局, 2004.
- [11] Rusty Russell, “Linux 2.4 NAT HOWTO,” January 2002.  
<http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html>
- [12] Rusty Russell and Harald Welte, “Linux Netfilter Hacking HOWTO,” July 2002.  
<http://www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO.html>
- [13] 白崎 博生, “実践セキュリティ”, ASCII, 2003.
- [14] Peter H. Salus, “UNIX の 1/4 世紀”, アスキー, 2000.
- [15] Mike Gancarz, “UNIX という考え方: その設計思想と哲学”, オーム社, 2001.
- [16] Linus Torvalds and David Diamond, “それがぼくには楽しかったから”, 小学館プロダクション, 2001.
- [17] 大竹龍史他, “標準テキスト CentOS 7 構築・運用・管理パーフェクトガイド”, SB クリエイティブ, 2017.
- [18] “ネットワークプロトコル 最強の指南書”, 日経 BP 社, 2017.
- [19] Eric Rescorla, “マスタリング TCP/IP SSL/TLS 編”, オーム社, 2003.
- [20] John Viega 他, “OpenSSL 一暗号・PKI・SSL/TLS ライブラリの詳細—”, オーム社, 2004.

## A サンプルリスト

### A.1 iptables 初期設定ファイル (iptables-sample.sh)

```
1 #!/bin/sh
2
3 PATH=/sbin:/bin:/usr/bin:/usr/sbin
4
5 ## 変数の定義
6 EXTERNAL_INTERFACE="enp1s0"      # 外側インタフェースの名前
7 DMZ_INTERFACE="enp2s0"           # DMZ インタフェースの名前
8 INTERNAL_INTERFACE="enp3s0"      # 内側インタフェースの名前
9
10 # 外側インタフェースの IP アドレス
11 IPADDR='ip addr show $EXTERNAL_INTERFACE | \
12 sed -e 's/^.*inet \([^ \/*\]*\).*$/\1/p' -e d'
13 # 内部ネットワーク・アドレス
14 INTERNAL_LAN='ip addr show $INTERNAL_INTERFACE | \
15 sed -e 's/^.*inet \([^ \/*\]*\).*$/\1/p' -e d'
16
17 # DMZ ネットワーク・アドレス
18 DMZ_LAN='ip addr show $DMZ_INTERFACE | \
19 sed -e 's/^.*inet \([^ \/*\]*\).*$/\1/p' -e d'
20
21 ANYWHERE="0.0.0.0/0"
22
23 ## 以下の設定を実行している間はパケットの転送を停止する
24 echo 0 > /proc/sys/net/ipv4/ip_forward
25
26 ## すでに設定されているルールを消去する
27 iptables -F
28 iptables -F -t nat
29
30 ## ポリシーの初期設定 -> match しない場合の扱い
31 iptables -P INPUT DROP
32 iptables -P OUTPUT DROP
33 iptables -P FORWARD DROP
34
35 ## ループバック・インタフェースの入出力を許可する
36 iptables -A INPUT -i lo -j ACCEPT
37 iptables -A OUTPUT -o lo -j ACCEPT
38
39 #####
40 ##
41 ## INPUT チェーンの設定 (デフォルト拒否)
42 ##
43
44 iptables -A INPUT -i $EXTERNAL_INTERFACE -p tcp -m state --state NEW -m tcp \
45 --dport 22 -j ACCEPT
```

```

46 iptables -A INPUT -i $INTERNAL_INTERFACE -p tcp -m state --state NEW -m tcp \
47     --dport 22 -j ACCEPT
48 iptables -A INPUT -i $INTERNAL_INTERFACE -p udp -m state --state NEW -m udp \
49     --dport 67 -j ACCEPT
50 iptables -A INPUT -i $INTERNAL_INTERFACE -p icmp -j ACCEPT
51 iptables -A INPUT -i $DMZ_INTERFACE -p udp -m state --state NEW -m udp \
52     --dport 67 -j ACCEPT
53 iptables -A INPUT -i $DMZ_INTERFACE -p icmp -j ACCEPT
54 iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
55
56
57 #####
58 ##
59 ## OUTPUT チェーンの設定（デフォルト拒否）
60 ##
61
62 iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p tcp -m state --state NEW -m tcp \
63     --dport 22 -j ACCEPT
64 iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p tcp -m state --state NEW -m tcp \
65     --dport 80 -j ACCEPT
66 iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p tcp -m state --state NEW -m tcp \
67     --dport 443 -j ACCEPT
68 iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p udp -m state --state NEW -m udp \
69     --dport 53 -j ACCEPT
70 iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p icmp -j ACCEPT
71 iptables -A OUTPUT -o $INTERNAL_INTERFACE -p tcp -m state --state NEW -m tcp \
72     --dport 22 -j ACCEPT
73 iptables -A OUTPUT -o $INTERNAL_INTERFACE -p icmp -j ACCEPT
74 iptables -A OUTPUT -o $DMZ_INTERFACE -p tcp -m state --state NEW -m tcp \
75     --dport 22 -j ACCEPT
76 iptables -A OUTPUT -o $DMZ_INTERFACE -p icmp -j ACCEPT
77 iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
78
79
80 #####
81 ##
82 ## FORWARD チェーンの設定（デフォルト拒否）
83 ##
84
85 iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p tcp \
86     -m state --state NEW,ESTABLISHED -m tcp --dport 22 -j ACCEPT
87 iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p tcp \
88     -m state --state NEW,ESTABLISHED -m tcp --dport 80 -j ACCEPT
89 iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p tcp \
90     -m state --state NEW,ESTABLISHED -m tcp --dport 443 -j ACCEPT
91 iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p udp \
92     -m state --state NEW,ESTABLISHED -m udp --dport 53 -j ACCEPT
93 iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p icmp -j ACCEPT

```

```

94 iptables -A FORWARD -i $EXTERNAL_INTERFACE -o $INTERNAL_INTERFACE \
95     -m state --state RELATED,ESTABLISHED -j ACCEPT
96 iptables -A FORWARD -i $INTERNAL_INTERFACE -o $DMZ_INTERFACE -j ACCEPT
97 iptables -A FORWARD -i $DMZ_INTERFACE -o $INTERNAL_INTERFACE \
98     -m state --state RELATED,ESTABLISHED -j ACCEPT
99
100 #####
101 ##
102 ## NAT の設定
103 ##
104
105 iptables -A POSTROUTING -t nat -s $INTERNAL_LAN -o $EXTERNAL_INTERFACE -j SNAT \
106     --to-source $IPADDR
107
108 #####
109 ##
110 ## 設定の保存
111 ##
112 #/etc/init.d/iptables save active
113
114 ## パケットの転送を開始する
115 echo 1 > /proc/sys/net/ipv4/ip_forward
116
117 exit 0

```

## A.2 WSGI サンプル (sample.wsgi)

```
1  #! /usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  import os
6  import io
7  import re
8  import textwrap
9
10 # CGI モジュールをインポート
11 import cgi
12 import cgitb
13 cgitb.enable()
14
15 localhost=False
16
17 # sqlite3 (SQL サーバ) モジュールをインポート
18 import sqlite3
19
20 # データベースファイルのパスを設定
21 app_dir = os.path.dirname(os.path.abspath(__file__))
22 dbname = os.path.join(app_dir, 'database.db')
23
24 # テーブルの作成
25 def createTable():
26     # データベース接続とカーソル生成
27     con = sqlite3.connect(dbname)
28     cur = con.cursor()
29     # テーブルの作成
30     create_table = 'create table if not exists users (id int, name varchar(64))'
31     cur.execute(create_table)
32     # コミット (変更を確定)
33     con.commit()
34     # カーソルと接続を閉じる
35     cur.close()
36     con.close()
37
38 createTable()
39
40 # 学生情報のデータベースへの登録
41 def registerStudent(std_id, std_name):
42     con = sqlite3.connect(dbname)
43     cur = con.cursor()
44     con.text_factory = str
45
46     # SQL 文 (insert) の作成と実行
47     sql = 'insert into users (id, name) values (?,?)'
```

```

48     cur.execute(sql, (std_id, std_name))
49     con.commit()
50
51     cur.close()
52     con.close()
53
54 # 登録情報一覧の取得
55 def getAllstudents():
56     # データベース接続とカーソル生成
57     con = sqlite3.connect(dbname)
58     cur = con.cursor()
59     con.text_factory = str
60
61     # SQL 文 (select) の作成と実行
62     sql = 'select * from users'
63     cur.execute(sql)
64     lists = cur.fetchall()
65
66     cur.close()
67     con.close()
68
69     return lists
70
71 # CSS ファイルなどを送信 (リファレンス WEB サーバ実行時のみ使用)
72 def serveFile(environ, start_response):
73     # 拡張子とコンテンツタイプ (必要に応じて追加すること)
74     types = {'.css': 'text/css', '.jpg': 'image/jpeg', '.png': 'image/png'}
75
76     # static ディレクトリ以下の指定ファイルを開いてその内容を返す.
77     # ファイルが無ければエラーを返す.
78     #
79     assert(re.match('/static/', environ['PATH_INFO']))
80     filepath = '{dir}/{path}'.format(dir=app_dir, path=environ['PATH_INFO'])
81     ext = os.path.splitext(filepath)[1]
82
83     if os.path.isfile(filepath) and ext in types:
84         try:
85             with open(filepath, "rb") as f:
86                 r = f.read()
87                 binary_stream = io.BytesIO(r)
88         except:
89             start_response('500 Internal Server Error', [('Content-Type', 'text/plain')])
90             return [b"Internal server error"]
91     else:
92         start_response('404 Not Found', [('Content-Type', 'text/plain')])
93         return [b"Not found."]
94
95     start_response('200 OK', [('Content-Type', types[ext])])

```



```

96     return binary_stream
97
98 # アプリケーション本体
99 def application(environ,start_response):
100
101     if localhost and re.match('/static/', environ['PATH_INFO']):
102         ''' リファレンス WEB サーバ実行時のみ
103             この wsgi スクリプトを実行しているカレントディレクトリ以下に
104             static というディレクトリを作って, default.css というファイルを置いておくと,
105             http://localhost:8080/static/default.css でファイルがダウンロードできる.
106         '''
107         return serveFile(environ,start_response)
108
109 # HTML (共通テンプレート)
110 tpl = textwrap.dedent('''
111 <html lang="ja">
112 <head>
113 <meta charset="UTF-8">
114 <title>WSGI テスト</title>
115 <link rel="stylesheet" href="/static/default.css">
116 </head>
117 {body}
118 </html>
119 ''')
120
121 # フォームデータを取得
122 form = cgi.FieldStorage(environ=environ,keep_blank_values=True)
123
124 if 'register' in form:
125     # 入力フォームで登録ボタンがクリックされた場合
126
127     # フォームデータから各フィールド値を取得
128     v1 = form.getvalue("v1")
129     v2 = form.getvalue("v2")
130
131     if v1 != '' and v2 != '':
132         # データベースへ学生の登録
133         registerStudent(int(v1), v2)
134         msg = '登録しました。'
135     else:
136         msg = '登録失敗：空になっている入力項目があります。'
137
138     # データベースの登録情報一覧の取得
139     students_list = getAllstudents()
140
141     # 一覧の HTML 形式への変換
142     list = ''
143     for row in students_list:

```

```

144         list += f'<li>{str(row[0])}, {row[1]}</li>\n'
145
146     bcontent = textwrap.dedent('''
147     <body>
148     <p>{message}</p>
149     <hr>
150     <div class="ol1">
151     <h2>一覧表示</h2>
152     <ul>
153     {slist}
154     </ul>
155     </div>
156     <a href="./sample.wsgi">登録ページに戻る</a>
157     </body>
158     ''').format(message=msg, slist=list)
159
160     else:
161         # デフォルトページ（入力フォーム表示）
162
163         # HTML（入力フォーム部分）
164         bcontent = textwrap.dedent('''
165         <body>
166         <div class=header>入力フォーム</div>
167         <div class="form1">
168         <form>
169         学生番号（整数） <input type="text" name="v1"><br>
170         氏名 （文字列） <input type="text" name="v2"><br>
171         <button type="submit" name="register">登録</button>
172         </form>
173         </div>
174         </body>
175         ''')
176
177         html = tpl.format(body=bcontent)
178         html = html.encode('utf-8')
179
180         # レスポンス
181         start_response('200 OK', [('Content-Type', 'text/html; charset=utf-8'),
182                                   ('Content-Length', str(len(html))) ])
183         return [html]
184
185
186
187 # リファレンス WEB サーバを起動
188 # ファイルを直接実行する（python3 sample.wsgi）と、
189 # リファレンス WEB サーバが起動し、http://localhost:8080 にアクセスすると
190 # このサンプルの動作が確認できる。
191 # コマンドライン引数にポート番号を指定（python3 sample.wsgi ポート番号）した場合は、

```

```
192 # http://localhost: ポート番号 にアクセスする.  
193 from wsgiref import simple_server  
194 if __name__ == '__main__':  
195     port = 8080  
196     localhost = True  
197     if len(sys.argv) == 2:  
198         port = int(sys.argv[1])  
199  
200     server = simple_server.make_server('', port, application)  
201     server.serve_forever()
```

## B 不正アクセスの手口・被害

最近では自宅でも光ファイバーや ADSL といった高速な通信回線を利用できるようになり、自分の PC をインターネットに常時接続することも少なくなってきた。また、みなさんが今後配属される研究室では、より多くのコンピュータをインターネットに接続していることだろう。

ここで問題となるのは、外部からの不正なアクセスを防ぐ方法である。携帯電話でときどきメールをチェックするのは異なり、コンピュータをインターネットに常時接続しておくことは、外部からの不正なアクセスを許す可能性を非常に高くする。ここでは、不正アクセスの手口と被害について紹介する。3 章の情報セキュリティ対策を学習する前に、事前知識としてぜひ一読してほしい。

なお、不正アクセスの手順などを具体的に説明しているが、**不正アクセスは犯罪である**ということをよく認識し、くれぐれも**実際にそれらを試してみようなどと思わないこと**。仮にそのような不正アクセスを行った場合、**大学を退学になるのはもちろんだが、法的にも裁かれ、被害によっては莫大な損害賠償を請求されることにもなる**。

### B.1 不正アクセスの被害

2019 年 3 月に、IPA（独立行政法人情報処理推進機構）が 2018 年に発生した情報セキュリティの事故・事件のうち、社会的に影響が大きかったと考えられる脅威のトップ 10 を公開している<sup>48</sup>。2016 年から、影響を受ける対象の違いから「個人」と「組織」という 2 つの分類で 10 大脅威を選出している（表 B.1）。

表 B.1：「情報セキュリティ 10 大脅威 2019」個人別・組織別順位

昨年 順位	個人	順位	組織	昨年 順位
1 位	クレジットカード情報の不正利用	1 位	標的型攻撃による被害	1 位
1 位	フィッシングによる個人情報等の詐取	2 位	ビジネスメール詐欺による被害	3 位
4 位	不正アプリによるスマートフォン利用者への被害	3 位	ランサムウェアによる被害	2 位
NEW	メール等を使った脅迫・詐欺の手口による金銭要求	4 位	サプライチェーンの弱点を悪用した攻撃の高まり	NEW
3 位	ネット上の誹謗・中傷・デマ	5 位	内部不正による情報漏えい	8 位
10 位	偽警告によるインターネット詐欺	6 位	サービス妨害攻撃によるサービスの停止	9 位
1 位	インターネットバンキングの不正利用	7 位	インターネットサービスからの個人情報の窃取	6 位
5 位	インターネットサービスへの不正ログイン	8 位	IoT 機器の脆弱性の顕在化	7 位
2 位	ランサムウェアによる被害	9 位	脆弱性対策情報の公開に伴う悪用増加	4 位
9 位	IoT 機器の不適切な管理	10 位	不注意による情報漏えい	12 位

(\*) クレジットカード被害の増加とフィッシング手口の多様化に鑑み、2018 年個人 1 位の「インターネットバンキングやクレジットカード情報等の不正利用」を①インターネットバンキングの不正利用、②クレジットカード情報の不正利用、③仮想通貨交換所を狙った攻撃、④仮想通貨採掘に加担させる手口、⑤フィッシングによる個人情報等の詐取、に分割している

被害に遭わないようにするためには、以下で説明する不正アクセスの手口や被害を理解することが最初の一步である。それに加えて、最新の脅威を知ることにより、年々巧妙化するセキュリティ・インシデントに対する心構えや適切な対策を講じることができるので、IPA のサイトに公開されている上記 10 大脅威に関する詳しい解説にも目を通してもらいたい。また、2016 年 2 月には、内閣サイバーセキュリティセンター（NISC）が、サイバーセキュリティに関する普及啓発活動の一環として、インターネット上のトラブルから身を守るための方法をわかりやすく解説した「ネットワークビギナーのための情報セキュリティハンドブック<sup>49</sup>」を作成・公開し、2017 年 7 月からは各種電子書籍での無料配信も開始しているため、そちらもあわせて読んでほしい。

<sup>48</sup>情報セキュリティ 10 大脅威 2019: <https://www.ipa.go.jp/security/vuln/10threats2019.html>

<sup>49</sup>ネットワークビギナーのための情報セキュリティハンドブック: <https://www.nisc.go.jp/security-site/handbook/>

### B.1.1 Web ページの改ざん

権限を持っていない人間によって情報が書き換えられることを改ざんという。Web ページの改ざんには大きく分けて次の二つのケースがある。

- Web サーバのホストに不正侵入され、ページが改ざんされる
- FTP や、CGI などのネットワークサービスが不正利用され、ファイルが置き換えられる

不正侵入や不正利用が起こる要因は、Web サーバのソフトウェアのバグ、CGI プログラムのバグ、その他のネットワークアプリケーションのバグ、遠隔ログインやファイル転送におけるアクセス制御の失敗、各種設定の誤り、など様々であり、Web ページの改ざんを防ぐには、これらの要因をできる限り少なくすることが望まれる。実際の侵入や改ざんには、クラッカーによる攻撃とコンピュータウィルスによるものがある。

2001 年 2 月～3 月には、H.U.C(Hacker Union Of China) を名乗る団体が、日本サイトへの攻撃予告をインターネット上のサーバに掲載し、それに関係すると思われる日本のサイトへの被害が続いた。2005 年 5 月には、PC や家電製品の価格比較サイト「価格.com」が不正アクセスにより改ざんされ、ウィルスを仕込まれる事件が起きた。最近では 2013 年 5 月下旬から、トヨタ自動車をはじめとする国内の企業・自治体などの Web サイトが相次いで改ざんされ、利用者の PC の状態によっては閲覧しただけでウィルス感染したり悪質サイトへ誘導されたりする可能性が高まった。Web ページが書き換えられると、その部署のセキュリティ意識、危機管理意識が問われ、場合によってはその部署が属する組織全体、我々の場合だと大学の信用を失墜してしまうことにもなりかねない。本学でも、2014 年 3 月に情報科学研究科の Web ページで使用していた CMS の脆弱性を狙った改ざんが発生し、長期間にわたって利用者に不便を強いることになった。情報技術の専門家集団として、このような事態を引き起こすことは決して許されないということを肝に銘じておく必要がある。

### B.1.2 踏み台

不正アクセスをした覚えがないにも関わらず、「貴方のサーバが当社のサーバに不正にアクセスしている」というクレームを受けたとする。この場合、サーバが「踏み台」にされた可能性が極めて高い。踏み台とは無関係な第三者のサーバを経由して、別のサーバに不正アクセスを試みる手段のことである。踏み台には、犯人の追跡を困難にするという特色もある。適切な情報セキュリティ対策を施していないと、このように被害者でありながら加害者になってしまう可能性がある。被害を被った企業からは、損害賠償請求など法的な手段により訴えられる恐れもある。

2003 年 6 月には、神戸大学で「休講掲示板システム」のサーバが踏み台にされ、米航空宇宙局 (NASA) のサーバに侵入を試みる不正プログラムが仕掛けられるという事件が起きている。最近では、オープンソースの CMS 「WordPress」や「Joomla!」の脆弱性が悪用され、踏み台にされるケースが発生した。本学でも、次項で述べる DDoS 攻撃を目的とした踏み台にされるインシデントがたびたび発生しており、注意が必要である。2000 年 2 月に施行された通称「不正アクセス禁止法」では、コンピュータの管理者は、不正アクセス行為から防御するための必要な措置を講ずるよう努力することが求められており、「知らなかった」「身に覚えがない」だけでは済まされなくなっている。

### B.1.3 サーバダウン

サーバダウンは、通称「DoS(Denial of Service) アタック」と呼ばれる方法によって引き起こされる。通常、サーバはアクセスしてきた要求に対し返答を行うが、それを逆手に取って、数万～数十万というサーバの許容量を超えるアクセス要求を行うことにより、サーバの稼働率を 100% にしてしまい、通常のサーバへのアクセスをさせないようにしてしまう手口である。そのような DoS アタックをされると、サーバが稼働しているコンピュータのリソースが減ってしまい、コンピュータがハングアップしてしまうこともある。複数のマシンから DoS アタックを試みたり (DDoS アタック:Distributed Denial of Service Attack)、踏み台を通して DoS アタックを仕掛けられることも多い。

2004 年 9 月から 2005 年 1 月にかけて、靖国神社の Web サイトが DDoS アタックを受けてニュースとなったほか、2004 年 11 月には大阪府のサーバがアタックを受け、約 2 時間 20 分にわたって同府の Web サイトにアクセスできないという状況になった。2013 年 3 月には、スパム対策組織「Spamhaus」をターゲットにした大規模な

DDoS 攻撃が発生し、その多くは外部からの再帰検索を許可しているネームサーバ（オープンリゾルバ）を悪用したものであった。

最近では、ルーターや監視カメラ、各種センシングデバイスなどインターネットに接続している IoT(Internet of Things) 機器を利用した DDoS 攻撃 (IDDoS) が大きな脅威になっている。2016 年 9 月 20 日には、「Mirai」と呼ばれるウィルスに感染した約 18 万台の IoT 機器が、著名なセキュリティジャーナリスト Brian Krebs 氏の Web サイトに対して約 620 Gbps の DDoS 攻撃を行っている<sup>50</sup>。

また必ずしも DoS 攻撃を意図しない行為としては、2010 年 5 月に岡崎市立中央図書館のホームページに対して、Web 上の文書や画像ファイル等を周期的に収集するプログラム (クローラ) を実行したことにより閲覧しにくくなる状況が発生し、プログラム作成者が逮捕されるという事件が起こった。図書館のシステムが古くもともと高負荷に耐えられない仕組みであったとは言え、機能停止やシステム損壊に至る事態を引き起こすことは絶対に避けなければならない。このようなプログラムを走らせた場合、サイトによっては自動的に一定期間の接続を拒否することもあり、たった一人の行為が大学全体に影響を与える可能性があることを十分に理解しておかねばならない。

#### B.1.4 情報の漏洩

情報の漏洩事件が続発している。2004 年 1 月には、ソフトバンク BB の加入者などの個人情報 450 万件が漏洩し、大きな社会問題となった。その後も事件は頻発しており、直近では、2013 年 5 月に、ヤフーが運営する「Yahoo! JAPAN」サービス用アカウントが最大 2200 万件（そのうちの約 149 万件はハッシュ化されたパスワードや秘密の質問を含む）が流出した。2013 年 7 月には、NTT コミュニケーションズが運営する「OCN」の ID 用メールアドレスとハッシュ化されたパスワードなど約 400 万件が流出し、LINE が運営する「NAVER アカウント」サービスの日本向けアカウント約 169 万件のメールアドレス、ユーザ ID、ハッシュ化されたパスワードが不正アクセスにより漏洩した。これらの事件から、情報を取り扱う側の姿勢が厳しく問われている。

大学の研究室という小さな組織であっても、多くの情報を保有している。たとえば、開発中のシステムの仕様書、評価データ、研究室メンバーの住所録といった個人情報、ユーザアカウント情報などである。それらの情報の取り扱いについては、明確な指針に従うべきである。たとえば、卒業生や退学者のユーザアカウントやパスワードの処置であるとか、研究データの持ち出しなどである。いくら物理的に強固なセキュリティ対策を施しても、利用する面でのセキュリティを考えておかなければ意味はない。2013 年後半には、デジタル複合機に保管された文書がインターネット上に公開されるという問題が相次いで発覚しており、2015 年 1 月には首都大学東京において NAS に保管された延べ 5 万人を超える個人情報が外部からアクセス可能な状態になるという問題が発生している。複合機や NAS といった機器に対してグローバル IP アドレスを付与し、ファイアウォールを設置せずインターネットに直接接続するという行為が如何に危険な状態であるかということをもっとよく理解する必要がある。ちなみに、名古屋大学では、情報連携統括本部の情報セキュリティに関するページ<sup>51</sup>の中で、情報セキュリティガイドラインとセキュリティポリシーが公開されているのでよく読んでおくこと。

#### B.1.5 メールを使った攻撃

メールを使った攻撃には、SPAM(スパム) メールやウィルス付メール、メール爆弾 (メールボム) などがある。SPAM メールとは、関係ない広告宣伝メールや勧誘メールを無差別に大量に送りつける行為あるいはその迷惑メール自体を言う。通常、SPAM メールは発信人が特定されないように、無関係なサーバを踏み台として中継利用される。犯人がメールアドレスを知らなくても、Web 上に公開されているメールアドレスを利用したり、架空のメールマガジンを使って自動的に収集したり、メールアドレスが「名前@プロバイダのドメイン名」になっている場合には辞書を使って簡単に類推することができるので注意が必要である。名古屋大学でも、2004 年 10 月中頃より、SPAM メールが急増し、同年 11 月にはメール配送サーバのダウンにまで至っている。

メール爆弾は、特別なプログラムを使い、数千～数万通以上のメールを一度に送りつけ、メールサーバの機能を麻痺させてしまうものである。本人にその気がなくてもメール爆弾と同じ行為になる可能性もある。例えば、卒論提出間際になって実験の手伝いやアルバイトの募集を、学内のメーリングリストを使って呼びかけたとする。メンバーが少なければ問題にならないかもしれないが、数百人、数千人規模に対して送ろうものなら、立派なメー

<sup>50</sup> 「IDDoS 攻撃」、IoT 機器から超弩級のサイバー攻撃: <http://itpro.nikkeibp.co.jp/atcl/column/16/120900297/121500001/>

<sup>51</sup> 名古屋大学情報連携統括本部 情報セキュリティ: <http://www.icts.nagoya-u.ac.jp/info/security.html>

ル爆弾犯である。アカウントの停止や、被害の規模が大きいなど下手をすると無期限停学といった厳重な処罰が待っている。こうなるともう卒業どころの話ではない。

### B.1.6 フィッシング詐欺

最近では、悪意の第三者が企業からのメールを装って偽のホームページへ誘導し、クレジットカード番号やID、パスワードといった情報を入力させて、不正に個人情報入手しようとする、フィッシング詐欺による被害が拡大している。最近では、ネットバンキングを運営する大手銀行、Apple、スクエア・エニックスを装ったフィッシング詐欺が発生している。手口も年々巧妙になっており、正規のサイトとまったく同じデザインの偽サイトに誘導してIDとパスワードを入力させたのち、正規サイトに転送して被害者が単に入力ミスをしたかのように装うような事例も存在する。

被害を防止するための対策としては、まず第一にホームページが本物であることの確認を行うことが挙げられる。URLが偽装されている可能性もあるので、URLだけを見て判断するのではなく、送られてきたメールのヘッダー情報を見たり、フィッシング詐欺の対策機能を持っているWebブラウザを利用すること、Webへのリンクを安易にクリックしないことも有効である。Twitterでは、文字数制限があることから短縮URLを利用することも多いが、便利な一方で悪意のあるWebページのURLを隠す効果もあることから、注意が必要である。短縮前のURLを表示してくれる機能を持ったブラウザを利用する、信頼できる人が紹介したURL以外は安易にクリックしない、といった自衛も重要である。

### B.1.7 パスワードクラックとトロイの木馬

パスワードクラックとは不正な手段を用いて、パスワードを盗み出すものである。使用されがちな単語や文字、人名辞書などを手掛かりに特殊なプログラムを用いてパスワードの解読を試みる方法が典型的な手口である。

トロイの木馬とは、一見有益な情報が得られるツールや面白いゲームのように見せかけているが、実は違う動作をするプログラムのことを言う。ウィルスのように他のコンピュータに感染することはないが、不用意に実行するとコンピュータ内のファイルが削除されるなどの被害を受ける。また、利用者が普段使っているユーザ名やパスワードを入力させるように仕掛けておき、パスワードを盗用する手口もある。

よく似た注意すべき事例として、意図しないソフトウェアのインストールがある。ソフトウェアのインストール時には、いくつかのステップを踏む過程でダイアログウィンドウが表示されるが、よく中身を読まずにボタンを押して先に進むことにより、「○○○○○をインストールする」のような項目にチェックが入っていることに気づかないまま、意図しないソフトウェアのインストールを許してしまうことが起こる。このようなソフトウェアは有用なものだけとは限らず、また、悪意のあるソフトウェアの場合は一度インストールしてしまったら最後、アンインストールが極めて複雑かつ困難なものも存在するので、利用者一人一人が十分気をつけるべきである。

利用者の入力を盗み取るという点では、クラウド型の入力支援ソフトにも注意が必要である。近年、PCやスマートフォンなど、多様なデバイスをさまざまな場所で使用することが多くなっているため、文字入力時の変換規則や辞書などのパーソナライズされた情報をクラウド上に保管することによって利便性を高める技術が盛んに提供されている。確かに、一度学習させた内容はデバイスに依存せず利用できることが望ましいし、多数の利用者の入力を統計的に処理することで変換品質の改善が期待されるというメリットも存在するが、自身の管理できない場所に置かれることから、最悪の場合、読みとられてしまうということを十分理解した上で、使い方に気をつけることが重要である。2013年12月には、中国検索大手百度製の日本語入力支援ソフトBaidu IMEが、利用者の入力した文字を無断で同社サーバに送信していたり、Android OS搭載の日本語入力アプリSimejiがクラウド機能が無効にしても入力した文字情報をサーバに無断送信していたということが起こっているので注意してほしい。

### B.1.8 セキュリティホール

侵入の要因となるセキュリティ体制やシステムの設定ミス、ソフトウェアのバグなどをセキュリティホールと呼ぶ。セキュリティホールの中でもソフトウェアに含まれる一部のプログラムは脆弱性と呼ばれ、この弱点を利用して攻撃するプログラムをエクスプロイトと呼ぶ。セキュリティホールは、サーバだけにあるわけではなく、Webブラウザやメーラーなどのクライアント側のソフトウェアにも存在する。



侵入者は、事前にセキュリティホールを探し、そこからエクスプロイトを利用して不正侵入を試みる。特に新しいセキュリティホールが公表された後は、このセキュリティホールを利用する者がいるため注意が必要である。そのため、サーバ管理者は、使用している OS やソフトウェアに新たにセキュリティホールが発生していないかを、日頃から注意しておく必要がある。2014 年 4 月に注意喚起された OpenSSL の HeartBleed 脆弱性や、Adobe Flash Player、Microsoft Internet Explorer など主要ソフトウェアの脆弱性を突いた攻撃も多数観測されており、情報収集と早急な対策によって被害を最小限にする努力が必要である。

### B.1.9 ポートスキャン

ポートとは、サーバとクライアントが通信を行うときに、データのやりとりを行う出入口の役割をするものであり、ポート番号と呼ばれる番号で表される。特別な設定をしない限り、やりとりするデータに応じて決められたポート番号が使用される。たとえば、Web ブラウザで Web サーバにアクセスするときは 80 番ポートを使い、メールを送信するときは 25 番ポートを使ってメールサーバとやりとりを行う。

ポートスキャンとは、サーバのどのポートが使用できるか (開いているか) を調べる行為である。ポートスキャン自体は不正なアクセスとは言い切れないが、不正アクセスをする前の準備段階として、どのホストが侵入しやすいかを調べ、ホストの OS の種類やバージョン、稼動中のサービスなどによりターゲットホストの脆弱性を判断するのである。クラッカーは、空いているポートを入口にして、攻撃を仕掛けてくる。

## B.2 不正アクセスの基礎知識

不正アクセスの究極の目的は、サーバの管理者権限 (root) を手に入れることとも言える。なぜなら、ルート権限を手に入れると、そのサーバ内においてはあらゆる行為が可能となるからである。例えばサーバ内の個人情報から機密情報まで、どんなディレクトリでもファイルでも自由自在にアクセスでき、書き換えも可能である。

### B.2.1 不正アクセスの手順

UNIX や Windows などのサーバを対象とした不正アクセスの基本的な流れを次に示す。

#### 1. ターゲッティング

ターゲッティングとは、攻撃対象のサーバの情報を得るために、サーバが提供する各種サービスから間接的かつ合法的に様々な情報を調べ、不正アクセスを行うための準備を行うことである。たとえば、ハンドルネームからメールアドレス、IP アドレスまで芋づる式に情報を収集したり、対象ホストの OS や管理者の技術力などを推測することができる。はじめに情報をできる限り多く集めておくことにより、不正アクセスのアプローチが容易になる。

#### 2. スキャン

スキャンとは、ネットワーク上にある様々なものを走査して情報を得る行為である。ターゲットとなるサーバの仕様などの情報を得るために必ず行うプロセスであり、その主な目的はターゲットの弱点となるセキュリティホールを探すことである。

#### 3. 間接的アタック

インターネット経由だけでなく、直接、ターゲットサーバの PC そのものにアクセスできる場合はこのステップをこなすことで比較的簡単にユーザ情報、ユーザ権限、ルート権限を得ることができる。

#### 4. ローカルアタック

攻撃対象のサーバのシェルアカウント<sup>52</sup>を持っている場合に、一般ユーザの権限からルート権限を奪取することを目的とする。アタック方法としては、Sniffer<sup>53</sup>、トロイの木馬<sup>54</sup>、セキュリティホールを突く攻撃などがある。対象サーバのソフトウェアのバージョン、OS の種類、OS のバージョンをスキャン作業によって事前に調べたことが役に立つ。シェルアカウントを持っていない場合は、リモートアタックを行う。

<sup>52</sup>シェル (shell) は、ユーザが入力したコマンドを解釈して OS に実行させるプログラム

<sup>53</sup>ネットワークのデータを盗聴するためのプログラム

<sup>54</sup>ネットワーク越しにコンピュータを悪意的に遠隔操作するためのプログラム

## 5. リモートアタック

他のサーバからネットワーク越しにサーバをアタックすることをリモートアタックと言う。リモートアタックは基本的にネットワークに接続された全てのホストに対して行えるので、汎用的なアタック方法と言える。一般的な攻撃対象サーバに侵入するためには、このステップでルート権限の奪取を狙う。ここでもスキャンで調べた情報が役に立つ。

## 6. ログ消去・ログ偽装

侵入したことはサーバのアクセスログ<sup>55</sup>に順次記録される。このログを消去しなければサーバ管理者に追跡される可能性がある。ゆえに、このログを消去することが侵入者にとって重要な作業となる。

## 7. 徘徊・改ざん

ルート権限を奪取し、ログを消去・改ざんできる環境が揃えば、サーバの不正アクセスに成功したことになる。これにより、そのサーバに存在する全ての情報にアクセス可能となる。

## 8. タイムスタンプの改ざん

ファイルの作成日時や最終アクセス日時などのタイムスタンプを改ざんすることによって、侵入が明らかになった際の時系列情報を混乱させ追跡を困難にする。

## 9. バックドア

バックドアは、一度ルート権限を奪取したサーバに再びアクセスする際、簡単にルート権限を得るために使う裏口のことである。これにより次回から簡単に侵入でき、踏み台<sup>56</sup>にすることもできる。最近では、VoIP 製品や Web カメラ<sup>57</sup> にバックドアが存在するといった問題も出ているので注意が必要である。

サーバ管理者に侵入がばれた場合、それをごまかすための簡単な方法として、次の最終手段が考えられる。

### ● コンソールアタック

エラーログを `/dev/console` に出力してコンソールをエラーメッセージで一杯にする。侵入を明らかにするというデメリットはあるが、どのみちばれているので追跡を困難にする意味では有効な手段と言える。

### ● ウィルス仕込む

ウィルススキャンを行うソフトウェアをアンインストールもしくは無効化したのち、忍び込ませたウィルスを起動させて感染させるという方法である。サーバ管理者に、侵入者の追跡よりもウィルス駆除に時間をかけさせることができる。

### ● すべてのハードディスクの削除

ハードディスク内のすべてのファイルを削除することで、攻撃対象サーバが別メディアにログを残したりバックアップをとっていない場合には、侵入者の追跡が困難になる。

### ● DoS アタック・DDoS アタック

DoS・DDoS アタックは、対象サーバをダウンさせるのに、最もポピュラーな攻撃方法と言える。DoS・DDoS アタックを行う場合、パケットやプロトコルの知識が必要となる。

このように不正アクセスは様々な工程を必要とするが、決してプログラムレベルだけで行われるものではなく、様々なツールを組み合わせることにより、スキルの低い人でも、工程や場面に合ったツールの使い方を理解していれば可能となるとところが怖いところである。

## B.3 不正アクセス対策の基本技術

不正アクセスの被害例や手口を知ったところで、情報セキュリティの基本技術について学ぼう。ここでは、様々な不正アクセス対策のうち、個人でも (やる気と努力があればなんとか) できる対策について、いくつか紹介する。

現在は Conficker や Stuxnet などがウィルスという枠を越え、タチの悪いワーム<sup>58</sup>として取りざたされている。これらは、これまでのウィルスの典型的な症状とされていたハードディスクの消去や、何かができなくなるといっ

<sup>55</sup>サーバに残されるアクセスの痕跡を記録するファイル

<sup>56</sup>特定のサーバにアタックする際に自分の身元をごまかすために利用するサーバのこと

<sup>57</sup>「中華ウェブカメラ」のセキュリティについて: <http://r00tapple.hatenablog.com/entry/2017/08/12/050252>

<sup>58</sup>OS やアプリケーションの弱点を利用して他の PC に拡散して自分自身を複製するように設計されたプログラムのこと

たことではなく、プライバシー度の高い情報を盗んでインターネット上に撒き散らす、産業制御システムを攻撃する、といった怖さがある。また、トロイの木馬 (Trojan) と呼ばれるものも存在する。これらに感染してしまうとそのトロイの木馬を設定した相手が感染者のコンピュータ内を自由に徘徊することが可能となり、ファイルの閲覧や操作などが簡単にできてしまう。

しかし、このような悪性コードは高度な技術やユーザが想像できない方法で伝播するわけではない。そして今後登場する悪性コードがどのような形態に変化するのかは未知数だが、以下の事項を注意すれば予防は可能である。

- 出所が明らかではないソフトウェアの使用を控える
- 使用するソフトウェアにセキュリティ脆弱性が発見されたらすぐにパッチを当てる
- 知らない人やよく分からない内容のメールは読まずに削除する
- システムのログインパスワードは簡単に類推できないものを使用する
- ドライブ全体の共有を控え、やむを得ない場合は「読み取り」権限で共有する
- 常に最新バージョンのウイルス対策プログラムを使用して定期的にスキャンする
- 最新のセキュリティ情報を熟知するようにする
- 重要なデータは定期的にバックアップを取っておく
- USB メモリなどの携帯ストレージ・デバイスを安易に貸し借りしない

特に、携帯ストレージ・デバイスについては、注意が必要である。なぜなら、まったく知らない者同士でそのようなデバイスの貸し借りをすることは少なく、多くの場合、自分の友人や同僚など信頼できる関係の中で当然のように起こり得るからである。また、PC と USB 経由で接続するデバイスとしては、USB メモリ以外にもデジタルカメラや音楽プレイヤーなどがあり、最近ではスマートフォンも感染源の対象となり得る。利用者本人には悪意がなくても、知らず知らずのうちにウィルスの運び屋をしているケースがある、ということ認識しておく必要がある。

これ以外にも多くの予防方法があるが、ユーザが少しでも関心をもって対処すれば、万一の事態が発生しても被害を最小限にとどめることができる。前述した「情報セキュリティ10大脅威 2016」では、江戸時代に坂本龍馬がまとめた「船中八策」にあやかり、情報セキュリティの8つの対策を「情報セキュリティ船中八策」(図 B.3)としてまとめており、2017 年度版はそのスマートフォン編 (図 B.4) を、2018 年度版はその IoT 機器 (情報家電) 編 (図 B.5) を示しているので参考にしてもらいたい。

一、ソフトウェアの更新  
～ 善は急げ ～  
二、ウイルス対策ソフトの導入  
～ 予防は治療に勝る ～  
三、パスワードの適切な管理  
～ 敵に塩を送ることのなきように ～  
四、認証の強化  
～ 念には念を入れよ ～  
五、設定の見直し  
～ 転ばぬ先の杖 ～  
六、脅威・手口を知る  
～ 彼を知り己を知れば百戦殆うからず ～  
七、クリック前に確認  
～ 石橋を叩いて渡る ～  
八、バックアップ  
～ 備えあれば憂いなし ～

図 B.3：情報セキュリティ船中八策

一、信頼できるサイトからインストール  
～ 触らぬ神に祟りなし ～  
二、アプリに許可する権限の確認  
～ 過ぎたるは猶及ばざるが如し ～  
三、脅威や手口を知る  
～ 彼を知り己を知れば百戦殆うからず ～  
四、認証の強化・データの暗号化・バックアップ  
～ 備えあれば憂いなし ～  
五、公衆無線 LAN の利用はリスクを理解  
～ 君子危うきに近寄らず ～  
六、パスワードを使い回さない  
～ 敵に塩を送ることのなきように ～  
七、OS・アプリの更新  
～ 善は急げ ～  
八、セキュリティソフトの導入  
～ 予防は治療に勝る ～

図 B.4：情報セキュリティ船中八策  
(スマートフォン編)

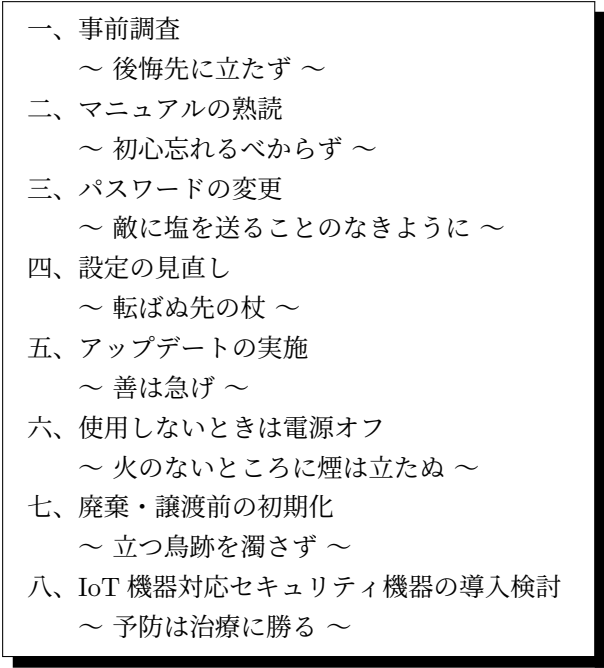
- 
- 一、事前調査  
～ 後悔先に立たず ～
  - 二、マニュアルの熟読  
～ 初心忘れるべからず ～
  - 三、パスワードの変更  
～ 敵に塩を送ることのなきように ～
  - 四、設定の見直し  
～ 転ばぬ先の杖 ～
  - 五、アップデートの実施  
～ 善は急げ ～
  - 六、使用しないときは電源オフ  
～ 火のないところに煙は立たぬ ～
  - 七、廃棄・譲渡前の初期化  
～ 立つ鳥跡を濁さず ～
  - 八、IoT 機器対応セキュリティ機器の導入検討  
～ 予防は治療に勝る ～

図 B.5：情報セキュリティ船中八策  
(IoT 機器 (情報家電) 編)

### B.3.1 コンピュータウィルス対策ソフトの導入

ウィルス対策ソフトは、コンピュータにウィルスが侵入したときに警告を発し、ウィルスを除去するソフトである。ウィルス対策ソフトは、一般に数千円程度で市販されている。また、パソコン購入時には既にインストールされている場合もある。ウィルス対策ソフトを導入することにより、ほとんどのウィルス被害を防ぐことが可能であるが、完璧かと言われればそうではない。というのも、悪意のある何者かによって新種のウィルスは作られ続けており、ウィルス対策ソフトをインストールしただけでは、古いウィルスにしか対応できないからである。ウィルス対策ソフトのメーカーは、新種のウィルスに対応するためのウィルス定義ファイル (パターンファイル) を用意し、インターネットを通じてファイルを公開しているので、ユーザは新種のウィルスにも対応できるよう常に最新の定義ファイルを利用する必要がある。ウィルス対策ソフトの中には、このファイルを自動的に更新するようなスケジュール機能を持つものもある。ただし、新しいウィルスの発生と、最新のウィルス定義ファイルの公開とは、少なからずタイムラグがあるので、ウィルス対策ソフトを入れているからといって安易に添付ファイルを開いたりすることは避けるべきである。

最近のウィルス対策として、ウィルスチェックプログラムをメールサーバに設定しておく方法がある。これは、電子メールが相手に配信される前に、メールにウィルスが添付されていないかどうかを調べ、もし、ウィルス付のメールがあればそのメールをサーバ上で破棄してしまうものである。

また、名古屋大学情報連携統括本部は、NICE に接続する端末のセキュリティ強化の一環として、Symantec Endpoint Protection のサイトライセンスを取得し、名古屋大学内の Windows・Mac 端末 (ただし、公的なものに限る) に提供している<sup>59</sup>。

### B.3.2 パッチプログラム

セキュリティホールが発生した場合、ソフトウェアのメーカーや制作者はそのセキュリティホールを修正するプログラムを公表する。これをパッチプログラムと言う。なお、パッチプログラムを修正プログラム、アップグレードプログラムなどと呼ぶ場合もある。セキュリティホールが公表された後は、そのセキュリティホールを利用した不正アクセスも発生するため、迅速にパッチプログラムをインストールする必要がある。

<sup>59</sup><http://w3serv.itc.nagoya-u.ac.jp/sitelicense/antivirus/> (学内からのみアクセス可)

Windows では、スタートメニューの中から、“Windows Update” を実行すれば、専用のサイトに接続して、現在のマシン環境にとって必要なパッチプログラムを提示してくれる。また、パッチプログラムのインストールまでを自動で行う設定もできる。

Linux では、たとえば apt-get や yum コマンドを使うことにより、システムを最新の状態に保つことができる。

### B.3.3 バックアップ/ログの管理

万一の場合に備えて、サーバ上のデータはバックアップをとっておくことが必要である。バックアップをとっておけば、何らかの原因でデータを損失してしまっても、バックアップをとった時点にまではデータを復旧させることができる。バックアップをとる際は、どのタイミングでバックアップをとるかを十分に検討し、プランニングすることが重要である。たとえば、毎朝きちんとバックアップをとっていても、毎回データを上書きしては、あまり意味がない。なぜなら、不正アクセスされ、データを改ざんされた後にバックアップをとっていた場合、データを改ざんする前には戻せないからである。一般には最低でも 3 世代のバックアップをとっておくことが必要だとされている。また、コンピュータの故障に備え、バックアップ先はバックアップ元のコンピュータとは違うコンピュータに保存することも検討する必要がある。

インターネットで利用されるサーバは、アクセスされた記録をログファイルとに記録する。ログファイルにはアクセスされた日時やファイル、アクセスしてきたコンピュータの IP アドレスなどが記録されている。このログファイルを分析することにより、アクセスが多い時間帯や人気のあるページなどを把握することができる。また、ログファイルは不正アクセスを試みられたときや、不正アクセスされたときの状況を調べる際にも用いられる。したがって、このログファイルもバックアップと同じように、ローテーションを組んで保存しておく必要がある。

### B.3.4 ファイアウォール

ファイアウォールとは、外部からの不正アクセスを防ぎ、内部ネットワークへの不正侵入を防ぐ技術である。ファイアウォールには様々な手法があるが、「パケットフィルタリング」と「アプリケーションゲートウェイ」が代表的な手法である。通常はこれらを組み合わせて、より強固なファイアウォールを構築する。

B.1.4 でも述べた通り、2013 年 11 月には、東京大学など国内の複数の大学で、複合機がインターネットから閲覧可能な状態になり、大量の印刷物を勝手に出力させたり、複合機内部に保存されたデータが盗み読まれるという問題が起きている。インターネットに公開する必要のない機器は、すべてファイアウォールの中に置き、外部との通信は厳しく制限するなどの対策が必要である。名古屋大学では、IP アドレス管理システム (IPDB) において、登録された IP アドレスでポートの公開申請が済んでいるもの以外の通信をすべて遮断するようになっている。

パケット (Packet) は、直訳すると小包という意味である。インターネットでやりとりするデータは、細かく分割されて送受信されるが、この細かく分割されたデータのことをパケットと言う。また、パケットにはヘッダと呼ばれる部分があり、ここに送信元や送信先のアドレス、接続するポート番号などの情報が保持されている。

外部から内部に流れてくるパケットのヘッダ部分を解析して、パケットを内部に流すかどうかを判断することをパケットフィルタリングと言う。パケットフィルタリングは、大きく分けて次の 2 種類がある。

- IP フィルタリング

ヘッダに許可しない送信元や送信先アドレスが記されていた場合、そのパケットは破棄する

- ポートフィルタリング

ヘッダに許可しないポート番号への接続要求があった場合、そのパケットは破棄する

アプリケーションゲートウェイもさまざまな実装方法があるが、プロキシサーバといわれるサーバを設置する方法が代表的である。プロキシ (proxy) とは、直訳すると代理という意味である。具体的には、外部ネットワークと内部ネットワークの間にプロキシサーバを置き、外部から流れるパケットを解析し、必要に応じてパケットを破棄する。また、内部ネットワークのクライアントコンピュータは、プロキシサーバを中継してインターネットなどの外部ネットワークにアクセスするため、クライアントコンピュータは外部ネットワークに直接には接続せず、セキュリティが高まる。



### B.3.5 パスワードポリシー

パスワードポリシーは、情報セキュリティポリシーに含まれるもので、パスワードに関するポリシーである。コンピュータやインターネットなどのネットワークを利用する際に、正規の利用者かを調べる最終手段である。そのため、情報セキュリティポリシーの中でも、最も重要なポリシーと言える。概ね、以下のことを検討する。

- パスワードは誰が決めるか (管理者か利用者本人か)
- パスワードの更新期間
- 以前使ったパスワードの使用可否
- パスワードの最短文字数
- パスワードに使用可能な文字種
- 使用禁止文字列 (誕生日や辞書にある単語など)

あまり強固なポリシーは、逆効果になることがある。たとえば、ポリシーが厳しすぎるため、パスワードが覚えきれず、パスワードを忘れてしまわないようにパスワードをメモした紙をディスプレイに貼ったりするケースはよくあることである。また、NIST（米国国立標準技術研究所）のコンピュータセキュリティ担当部門は、上記にも挙げているパスワードの更新期間について、「ユーザが攻撃を受けたという証拠に基づいて変更する場合を除いて、パスワードの定期変更をするべきではない」と従来のセキュリティの常識とは異なる見解を示している。難しいパスワードを定期的に作り直して覚えるという行為にそもそも無理があり、あまりしつこくシステム側が定期的な変更を求めると、ユーザが簡便なパスワードに切り替えたり、前のパスワードの一部を変更しただけの安易なものに変更したりすることによって、かえってリスクが高くなることを念頭に置いたものと考えられる<sup>60</sup>。

個人がパスワード設定で最低限気をつけるべきことは、上記の最後の3つである。すなわち、自分の誕生日や辞書に載っている単語・人名などを避け、英数字と記号を組み合わせた8文字以上のパスワードを設定するということである。たとえば、パスワードクラック (パスワード解析) ソフトを使うと、6文字程度の単語や人名を使ったパスワードはものの数分で破られてしまう。しかし、そのようなパスワードが覚えにくいものであるとするならば、文字種の組み合わせに腐心するよりもむしろシンプルに単語を4つ並べたパスワードの方が (文字数が多い分) 破りにくいのも事実である<sup>61</sup>。

また最近では、インターネットショッピングをはじめとする Web サービスなどで、パスワードを設定する機会が多くなったことから、ID やパスワードの使いまわしが多く行われており、それを狙った攻撃 (リスト型アカウントハッキング) も起きている。システムの脆弱なサイトから入手した ID とパスワードを使って、手当たり次第に他のサイトでログインを試行するという手口である<sup>62</sup>。

ID とパスワードの使いまわし自体は、必ずしも悪いとは言いきれない。というのも、利用するサービスごとにすべて異なる ID とパスワードを設定するというのは現実的でないし、管理が難しくなって結局どこかに書き留めてしまっただけは元も子もないからである。重要なのは、情報漏洩が仮に起きてしまった場合を想定し、そのサイトで扱っている情報の重要度に応じてパスワードの複雑さ (文字数・文字種) を決めるということである。また、どうしても覚えられない場合のあまりお勧めしない最終手段としては、パスワードそのものではなく、自分にしか分からないヒントを書き留めておくという方法もあるが、個人の自己責任で行う必要がある。

<sup>60</sup>NIST Special Publication 800-63B Digital Identity Guidelines. <https://pages.nist.gov/800-63-3/sp800-63b.html>, 2017.6.

<sup>61</sup>「あのパスワード規則、実は失敗作だった」: <http://blogos.com/article/239771/>, 2017.8.9.

<sup>62</sup>2014 年 7 月に起きた LINE アカウントの乗っ取りについてもこの手口が疑われている。

## B.4 不正アクセスされた場合の対処方法

サーバに不正アクセスや不正侵入を許してしまった場合の基本的な対処方法は次の通りである。

### 1. コンピュータの電源を切る

まず、不正アクセスの内容によってコンピュータの電源を切るかどうかの判断が必要になる。しかし、稼働中のコンピュータの電源を切ることは、ハードディスクを含む機器の故障リスクを高めるため、不正アクセスされた原因の追及を困難にする場合がある。一方で、ネットワークからの切り離しを先に実施してしまうと、不正アクセスに気づかれたと判断され、不正アクセスの証拠をはじめ、下手をするとハードディスクの中身すべてを自動消去される恐れがある。重大犯罪のようなケースでは、ハードディスクの故障リスクよりも不正アクセスの証拠を優先し、コンピュータの電源を強制的に切ることがしばしば行われる。

### 2. コンピュータをネットワークから切り離す

サーバが稼働したままの場合、そのサーバにアクセスしたクライアントにまで被害が及ぶ可能性がある。そのため、まずサーバをネットワークから切り離し、二次的な被害を防ぐことを考える。具体的にはコンピュータからネットワークケーブルを外す。

### 3. 被害状況を把握

どういう被害をどの状態まで受けたかを分析する。また、不正にネットワークにアクセスした侵入者は、再度、侵入しやすいように、バックドアを残しておく場合があり、注意が必要である。不正アクセスを受けたコンピュータのハードディスクを取り外し、書き込みできない状態で他のコンピュータに接続して中身を確認する必要がある。

### 4. 不正侵入された原因の調査

アクセスログは参考にはなるが、ログも改ざんされている場合もあるので 100%信用することはできない。

### 5. 復旧処理

復旧処理の手順はケースバイケースである。一般的には、被害にあう前の状況においての最新のバックアップファイルの状態に戻し、そこから必要なパッチプログラムをインストールして復旧する。ただし、どの時点で不正侵入されたかわからない場合は、バックドアを仕掛けられている可能性があり、OS のインストールから行う必要がある。重大犯罪のようなケースでは、警察から証拠提出を求められることもあるため、不正アクセスの証拠としてハードディスクは保管し、新たに用意したハードディスクに OS をインストールする必要がある。

## B.5 コンピュータウィルスが侵入したときの対処方法

ウィルスは次々と新種が作成されているため、ウィルス定義ファイルが公表される前にウィルスが流行し、コンピュータに感染してしまう場合がある。また、最近ではステルス型ウィルスといい、ウィルス対策ソフトに反応しにくい新手のウィルスも見つかっている。ウィルスに感染してしまった場合には、おおむね次の手順をとる。

### 1. コンピュータをネットワークから切り離す

ウィルスが感染したコンピュータがネットワークにつながっている場合は、直ちにネットワークから切り離す。具体的には、ネットワークケーブルを抜く。

### 2. メールサーバの停止

メールを媒介して伝染するウィルスの場合、メールソフトのアドレス帳にあるメールアドレスに無差別にウィルス付メールを送信するものがある。この場合は、内部だけでなく外部にもウィルス付メールを送信してしまう可能性がある。これを防ぐために、ウィルスが沈静化し、完全に駆除されるまでメールサーバを止めることも検討する。



ウィルス等に感染したコンピュータを発信源として、迷惑メールを大量に送信することはネットワーク管理上の問題となっており、組織単位での迷惑メールブラックリスト入りなど、他の利用者に迷惑をかけることにもつながる。名古屋大学では、本対策として 2017 年 9 月 19 日より全学ファイアウォールにて OP25B<sup>a</sup> を実施し、TCP 25 番ポートを用いたメール学外送信（学内→学外）を遮断している。

<sup>a</sup>OP25B: Outbound Port 25 Blocking. 迷惑メールの送信には、通常、ウィルスに感染したコンピュータから直接学外のメールサーバに TCP 25 番ポートで接続して迷惑メールを送りつける方法が使われる。そこで、同ポートを塞ぐことによってウィルス感染を原因とした迷惑メールの大量送信を抑える効果が期待される。

### 3. 定義ファイルの更新

利用しているウィルスソフトの定義ファイルを更新する。感染状況を取りまとめ、全台のコンピュータで定義ファイルの更新を行う。全台に処置が完了したことが確認できるまで、ネットワークは再開しないようにする。

### 4. 復旧作業

全台のコンピュータで定義ファイルを更新し、ウィルスの駆除が完了したことを確認したあとにネットワークの利用を再開する。その後、外部パケットの監視などにより感染の継続が疑われる（ウィルス対策ソフトによる除去が難しい）場合には、OS の再インストールを実施する。