

目次

1 OpenSSH (1).....	4	4.4.1 システムでの転送処理.....	52
1.1 SSHとは?.....	5	4.4.2 ユーザーによる転送処理.....	54
1.1.1 リモートログイン.....	5	5 メールサーバー (3).....	56
1.1.2 SSH プロトコル.....	5	5.1 POP サーバーの役割.....	57
1.1.3 OpenSSH.....	6	5.1.1 メールボックス.....	57
1.2 OpenSSH クライアント.....	7	5.2 Dovecot を使った POP サーバーの起動.....	59
1.2.1 SSH クライアントの利用.....	7	5.2.1 Dovecot のインストール.....	59
1.2.2 ホスト間認証.....	7	5.2.2 Dovecot の設定.....	59
1.2.3 UNIX パスワード認証.....	8	5.2.3 Dovecot の起動.....	60
1.3 OpenSSH サーバー.....	10	5.3 メールクライアントの導入.....	61
1.3.1 SSH サーバーの起動.....	10	5.3.1 Linux 上でのメールクライアント.....	61
1.3.2 SSH サーバーの設定.....	12	5.3.2 Sylpheed の起動と設定.....	61
1.3.3 SSH サーバーへのアクセス制限.....	14	6 NFS.....	66
1.4 確認問題.....	16	6.1 NFS の概要.....	67
2 OpenSSH (2).....	18	6.1.1 NFS の仕組み.....	67
2.1 公開鍵暗号認証.....	19	6.1.2 NFS サーバープログラム.....	67
2.1.1 暗号化とは.....	19	6.2 エクスポートの設定と NFS マウント.....	69
2.1.2 公開鍵暗号とは.....	19	6.2.1 エクスポートの定義.....	69
2.1.3 公開鍵暗号を用いた認証.....	21	6.2.2 エクスポート設定の反映と確認.....	69
2.2 OpenSSH における公開鍵暗号認証.....	22	6.2.3 NFS マウント.....	70
2.2.1 鍵の生成.....	22	6.2.4 NFS アンマウント.....	71
2.2.2 SSH サーバーへの公開鍵の登録.....	24	6.2.5 エクスポートオプション.....	72
2.2.3 公開鍵認証を用いたログイン.....	24	6.3 NFS マウントにおけるユーザー権限.....	74
2.2.4 SSH サーバー側の設定.....	25	6.3.1 UID、GID によるマッピング.....	74
2.3 安全なファイル転送.....	26	6.3.2 root ユーザーのマッピング.....	74
2.3.1 scp コマンド.....	26	6.3.3 その他のエクスポートオプション.....	75
2.3.2 sftp コマンド.....	27	6.4 確認問題.....	76
2.4 確認問題.....	28	7 FTP.....	78
3 メールサーバー (1).....	30	7.1 オープンソースソフトウェアの利用.....	79
3.1 メール配信の仕組み.....	31	7.1.1 パッケージの入手.....	79
3.1.1 メールの作成と送信.....	31	7.1.2 tarball の展開とドキュメントの確認.....	82
3.1.2 メールサーバーでの配送.....	31	7.1.3 コンパイル準備.....	82
3.1.3 メールの受信.....	32	7.1.4 コンパイルとインストール.....	84
3.2 メールサーバーの役割.....	33	7.2 FTP サーバー.....	85
3.2.1 ローカル配送とリモート配送.....	33	7.2.1 ProFTPD の基本的な設定.....	85
3.2.2 メールの中継と制限.....	33	7.2.2 ProFTPD の起動・停止.....	86
3.2.3 宛先と名前解決.....	35	7.2.3 ホストアクセス制限.....	87
3.2.4 メールの受信.....	35	7.2.4 ディレクトリ制限 (chroot).....	88
3.3 メールの構造.....	37	7.2.5 ユーザー制限.....	89
3.3.1 エンベロープとメール本体.....	37	8 システム管理 (1).....	92
3.3.2 ヘッドとメール本文.....	37	8.1 システムログ.....	93
3.4 電子メールのプロトコル.....	39	8.1.1 システムログとは.....	93
3.4.1 SMTP.....	39	8.1.2 ログのセレクト.....	94
3.4.2 POP.....	40	8.1.3 ログのアクション.....	95
4 メールサーバー (2).....	42	8.2 ジョブスケジューリング.....	97
4.1 Postfix.....	43	8.2.1 cron によるジョブの定期実行.....	97
4.1.1 Postfix の概要.....	43	8.2.2 at によるジョブの予約.....	98
4.1.2 Postfix の設定.....	43	8.3 NTP.....	100
4.2 Postfix の利用.....	46	8.3.1 NTP サーバー.....	100
4.2.1 Postfix の起動と停止.....	46	8.3.2 ntpdate.....	101
4.2.2 メールの配送とメールボックス.....	47	8.3.3 ntpd.....	101
4.2.3 Postfix のログ.....	49	8.4 オンラインマニュアル.....	103
4.3 Postfix の運用管理.....	50	8.4.1 オンラインマニュアルとは.....	103
4.3.1 キューの確認.....	50	8.4.2 オンラインマニュアルの利用.....	103
4.4 転送処理.....	52	8.4.3 オンラインマニュアルの構成.....	104

8.4.4 関連コマンド.....	105	10.2.2 コンソールの横取り.....	127
8.4.5 info コマンド.....	106	10.2.3 ネットワークからの侵入.....	129
8.4.6 その他の情報源.....	107	10.2.4 ソフトウェアのアップデート.....	129
9 システム管理(2).....	110	10.2.5 ソフトウェアの稼働状況.....	130
9.1 システムの運用状況の確認.....	111	10.3 パケットフィルタリング.....	131
9.1.1 CPU 負荷の監視.....	111	10.3.1 パケットフィルタリングとは.....	131
9.1.2 メモリ利用の確認.....	111	10.3.2 パケットフィルタリングの前提.....	132
9.1.3 リアルタイムの監視.....	112	10.4 iptables コマンド.....	133
9.1.4 ディスクの利用状況の確認.....	113	10.4.1 パケットの経路.....	133
9.2 冗長性のあるシステム構成の構築.....	115	10.4.2 iptables の開始.....	133
9.2.1 RAID.....	115	10.4.3 フィルタリングの方針.....	135
9.2.2 LVM.....	116	10.4.4 パケットフィルタリングの設定.....	137
9.2.3 冷却ファンや電源の二重化.....	117	10.5 確認問題.....	140
9.3 バックアップ.....	118	付録 確認問題 解答例.....	142
9.3.1 バックアップの対象.....	118	第1章.....	143
9.3.2 バックアップの種類.....	118	第2章.....	150
9.3.3 XFS でのバックアップとリストア.....	119	第3章.....	153
10 セキュリティ入門.....	124	第4章.....	154
10.1 セキュリティとは.....	125	第5章.....	161
10.1.1 守るべきもの.....	125	第6章.....	165
10.1.2 セキュリティを犯すもの.....	125	第7章.....	168
10.1.3 セキュリティ向上の方法.....	126	第8章.....	175
10.2 セキュリティ対策.....	127	第9章.....	178
10.2.1 物理構成に対するセキュリティ.....	127	第10章.....	180

1 OpenSSH (1)

1.1 SSHとは？

1.1.1 リモートログイン

通常はコンピューターを目の前にして操作しますが、実際のサーバー運用ではネットワークを介し、物理的に離れた場所にあるコンピューターにログインして操作を行うことが一般的です。これはセキュリティの観点からデータセンターと呼ばれる特別な施設にサーバーを設置しているためです。

物理的に離れたサーバーへ、ネットワークを介してアクセスすることをリモートログインといいます。リモートログインには長い間 TELNET というプロトコルが利用されてきましたがデータを素のまま平文(ひらぶん)で流してしまい、情報を盗聴される危険性がありました。特にユーザー名、パスワードといった情報など漏れるとサーバーが乗っ取られる危険があります。

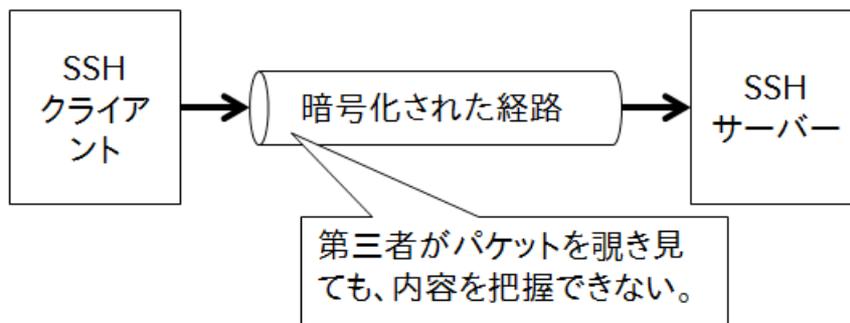
そこで、安全な通信を実現するために次の2つの要件が必要となります。

- たとえデータが盗聴されても、そのままでは利用できないこと
- 確実なユーザーとホストの認証(本人確認)

そこで「暗号化」が必要となります。暗号化通信により盗聴を無力化でき、認証情報も安心して送信でき、上記の2つの要請を満たすことができます。ここではもっとも普及している SSH について解説します。

1.1.2 SSH プロトコル

TELNET によるリモートログインは、データが暗号化されず、そのまま送信されます。このため、重要な情報が盗聴される危険性があります。そこで通信データを暗号化し、盗聴されても内容が漏れないようにする SSH (Secure Shell) が開発されました。



SSH は、クライアント・サーバー方式で実現されるサービスです。サーバとクライアント双方で暗号化をすることで、通信経路に流れる情報は全て暗号となり盗聴に強くなります。

1.1.3 OpenSSH

SSH は、1995 年当時ヘルシンキ工科大学の研究者だった Taru Ylönen (タル・ヨーネン) により開発されました。Ylönen は SSH を 7 月にフリーの OSS として公開しましたが、世界中から、メンテナンスやサポート依頼など、反響があまりに多かったことから、同年 12 月、SSH Communications Security 社を立ち上げ、商用化に至ります。この流れは IETF とも連動し、現在では RFC として制定されています。現在 OSS で使用されている中で、主流になっている SSH は、OpenBSD プロジェクトが開発した OpenSSH で、後に様々なプラットフォームに移植され、現在では多くの OS でサポートされています。

CentOS では SSH を利用するためには、以下のパッケージが必要となります。

パッケージ名	概要
openssh	OpenSSH の共通機能 (ssh-keygen)
openssh-clients	OpenSSH クライアントプログラム
openssh-server	OpenSSH サーバープログラム
openssl	OpenSSL 共通機能 (openssl コマンド)
openssl-libs	OpenSSL ライブラリ

OpenSSL は、SSH 通信で用いる暗号化を他のアプリでも利用できるよう汎用化したものです。

【練習】

1. rpm コマンドを利用して、既にインストールされている openssh 関連のパッケージを確認します。
2. インストール済みの openssh パッケージのバージョンを調べます。

1.2 OpenSSH クライアント

1.2.1 SSH クライアントの利用

SSH サーバーに接続するためには、SSH クライアントプログラムとして、OpenSSH パッケージの `ssh` コマンドを利用します。数々のオプションがサポートされていますが、基本的な用法は以下の通りです。

```
ssh ユーザー名@接続先のホスト
```

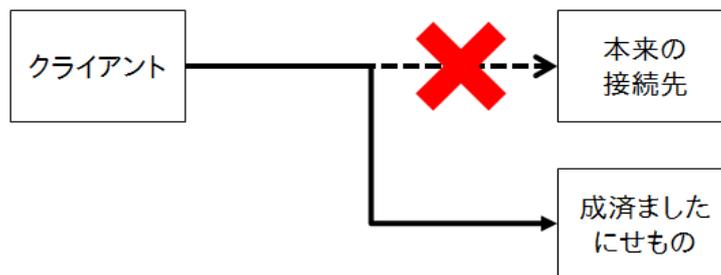
接続先のホストをホスト名または IP アドレスで指定します。サーバー側ホストのログインユーザーを指定するためには「ユーザー名@」を記述します。省略するとクライアント側カレントユーザー名と同名のサーバー側ユーザーとしてログインすることになります。

例: 10.20.142.6 のホストにユーザー student でログインする

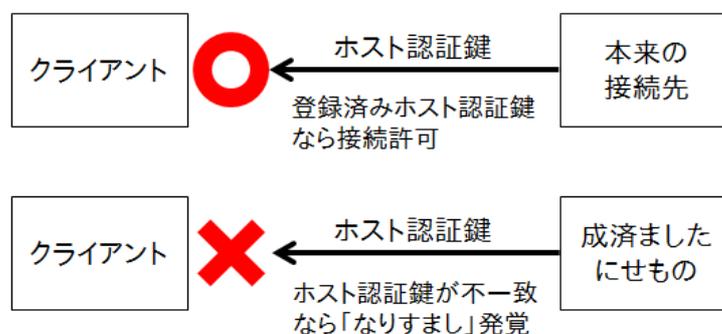
```
$ ssh student@10.20.142.6
```

1.2.2 ホスト間認証

通信を暗号化しても完璧ではありません。例えば接続先のホストになりすまし、重要なデータを奪う「なりすまし」が考えられます。



「なりすまし」の被害を回避するには、接続先が本当に意図したサーバーかどうかを確認する必要があります。SSH は初めて接続した際に、サーバーからホスト鍵（公開鍵）という情報が送られ、クライアントではこれを記録し 2 回目以降に突合することで、同じサーバーかどうかを判定しています。これをホスト間認証と呼びます。



ある SSH サーバーに対して、ssh コマンドを使用して通信を開始する場合、初回のログインにおいては、以下のようなメッセージが表示されます。

```
$ ssh student@10.20.142.6
The authenticity of host '10.20.142.6 (10.20.142.6)' can't be established.
ECDSA key fingerprint is ad:55:c3:5c:ca:11:3d:46:53:81:66:f2:01:d2:b4:fe.
Are you sure you want to continue connecting (yes/no)?
```

これは、クライアント側に登録されていないホスト間認証用鍵が、サーバー側から送られてきたことを示し、このホストを信用するかどうか、ユーザーに確認をとるメッセージです。

接続作業を続けるか否かという問い合わせに、「yes」と入力し作業続行を指示すると、以下のようなメッセージが表示されます。

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.20.142.6' (ECDSA) to the list of known hosts.
```

これは、接続先のホストの認証鍵がクライアント側に登録され、次回からは、この認証鍵を使ってサーバー側のペアの鍵をチェックすることを示しています。登録された認証鍵は、ユーザーのホームディレクトリにある「.ssh」ディレクトリ内の「known_hosts」ファイルに順次収容されていきます。

```
$ cat ~/.ssh/known_hosts
10.20.142.6 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAA
BBBNEZu4D7+I5aSF/uMnfeEzy/YA8o8r0tD/LoyxbA1NFzg2WRG6ZUB59CJ69No7ooBB2bvMVX3pmtSK4ENrEbJNI=
```

このファイルの形式は次の通りです。

```
ホスト名または IP アドレス   SSH 暗号化方式(アルゴリズム)   認証鍵のデータ
```

1ホストにつき1行として、全てのホスト認証鍵が記録されます。

1.2.3 UNIX パスワード認証

SSH を用いた通信は、サーバーのホスト認証が終了した後、ユーザー認証が行われます。認証方法は複数あり、シンプルな方法として、ローカルマシンへのログインと同様に、/etc/passwd (/etc/shadow) を用いた UNIX パスワード認証があります。

例: 10.20.121.6 へのログイン (2回目以降)

```
$ ssh student@10.20.121.6
student@10.20.121.6's password: [パスワードを入力]
Last login: Wed May 6 22:50:10 2017 from h006.s142.la.net
[student@h006 ~]$
```

ログイン時に求められるパスワードは、ログイン先ホストのユーザーのものであることに注意してください。サーバー側のホストに登録されているユーザーアカウントでログインします。

[練習]

1.3 OpenSSH サーバー

1.3.1 SSH サーバーの起動

OpenSSH のサーバープログラムは、`openssh-server` パッケージによりインストールされます。実行プログラムは「`/usr/sbin/sshd`」であり、`sshd` デーモンが SSH サーバーの役割を担います。`sshd` に関する設定ファイルは「`/etc/ssh/sshd_config`」で、デフォルトの設定は UNIX パスワード認証が可能になっています。

`ssh` を起動するには、`root` ユーザーにて `systemctl` コマンドを用います。

```
# systemctl start sshd
```

停止するには、引数に `stop` を指定し、再起動(停止したのち、起動)は `restart` を指定します。

```
# systemctl stop sshd
# systemctl restart sshd
```

また `status` により稼働状況を表示します。

```
$ systemctl status sshd
●sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset:
   enabled)
   Active: active (running) since 火 2017-07-28 08:25:36 JST; 34min ago
 Main PID: 1072 (sshd)
   CGroup: /system.slice/sshd.service
           └─ 1072 /usr/sbin/sshd -D

7月 28 08:25:36 h006.s142.la.net systemd[1]: Starting OpenSSH server daemon...
7月 28 08:25:36 h006.s142.la.net systemd[1]: Started OpenSSH server daemon...
7月 28 08:25:36 h006.s142.la.net sshd[801]: Server listening on 0.0.0.0 po...
7月 28 08:25:36 h006.s142.la.net sshd[801]: Server listening on :: port 22.
Hint: Some lines were ellipsized, use -l to show in full.
```

【練習】

1. ps コマンドで sshd が起動していることを確認します。
2. ss または netstat コマンドを利用して、sshd の待ち受けポートが 22/tcp で、LISTEN になっていることを確認します。
3. systemctl コマンドを使って、sshd を停止します。
4. ps コマンドを利用して sshd が動作していないことを確認します。併せて、22/tcp が LISTEN されていないことを確認します。
5. systemctl コマンドを使用して、sshd を起動します。
6. ps コマンドで sshd が動作していることを確認します。併せて、22/tcp が LISTEN されていることを確認します。
7. ユーザー student になり、localhost に対して ssh コマンドを実行し、SSH でログインします。
8. ログアウトします。
9. ホームディレクトリの .ssh/known_hosts ファイルを参照し、自分のホストの認証鍵が登録されていることを確認します。

1.3.2 SSH サーバーの設定

OpenSSHでのサーバーデーモン sshd の設定は、設定ファイル sshd_config で行います。CentOS では、ディレクトリ/etc/ssh/以下に配置されています。

```
#      $OpenBSD: sshd_config,v 1.93 2014/01/10 05:59:19 djm Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.
    (省略)
# If you want to change the port on a SELinux system, you have to
tell
# SELinux about this change.
# semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
#
#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
    (以下略)
```

設定ファイルの書式は以下のようになっています。「#」は、それ以降をコメントとし、動作に影響しません。

```
# コメント
設定名      設定値
```

インストール直後の設定ファイルには、設定行が「#」でコメントアウトされている部分が数多く見られますが、これはデフォルトの設定値を示しています。

例えば、sshd サーバーがクライアントからの要求を受け付ける TCP ポートに定義する「Port」は、

```
# Port 22
```

と記述されています。これはコメントアウト(無効化)されていますが、デフォルト値が 22 であることを意味しています。

以下に主な設定をまとめます。

設定名	解説(規定値)
Port	sshd が要求を Listen するポートのポート番号(22)。
Protocol	sshd がサポートする SSH プロトコルのバージョン(2)。
HostKey	クライアント側に送信するホスト認証鍵を納めたファイル名。通常は変更しない(/etc/ssh/ssh_host_rsa_key などアルゴリズムごとに存在)。
LoginGraceTime	クライアントの認証の猶予時間[秒]。サーバに接続してからこの時間放置すると自動的に接続が切断される(2m)。
PermitRootLogin	root ユーザーによるログインの可否。 特に理由がない限り、拒否するのが望ましい(yes)。
PasswordAuthentication	UNIX パスワード認証の可否。(yes)。
PermitEmptyPasswords	空のパスワードによる認証の可否。通常は許可しない(no)。

UNIX パスワード認証を許可するためには「PasswordAuthentication」を「yes」に指定する必要があります。デフォルトの設定は「yes」が指定されています。

/etc/ssh/sshd_config の設定を変更した場合は、sshd を再起動して、新しい設定を有効にする必要があります。

```
# systemctl restart sshd
```

なお sshd を再起動しても、使用中の接続は切断されません。

【練習】

1. sshd の設定ファイルの絶対パスを確認して、設定ファイルの内容を参照します。
2. sshd の設定において、「root ユーザーによるログインを許可するかどうか」を指定している設定項目を見つけ、設定値を確認します。
3. 自分のホストの SSH サーバーに対して、root ユーザーでのログインを試み、成功することを確認します。
4. sshd の設定ファイルを編集し、「root ユーザーによるログインを禁止する」よう設定を変更します。
5. 新しい設定を有効にするため、sshd を再起動します。
6. 再度、自分のホストの ssh サーバーに対して、root ユーザーでのログインを試みます。

1.3.3 SSH サーバーへのアクセス制限

デフォルトの設定では、全てのホストからの SSH サーバーへのアクセスが許可されています。SSH による通信は信頼できるホストからのみ可能ですので、通常はこのままの設定で構いません。特に制限を行いたい場合は、`/etc/hosts.allow` ファイルと `/etc/hosts.deny` ファイルを用いて、設定を行います。

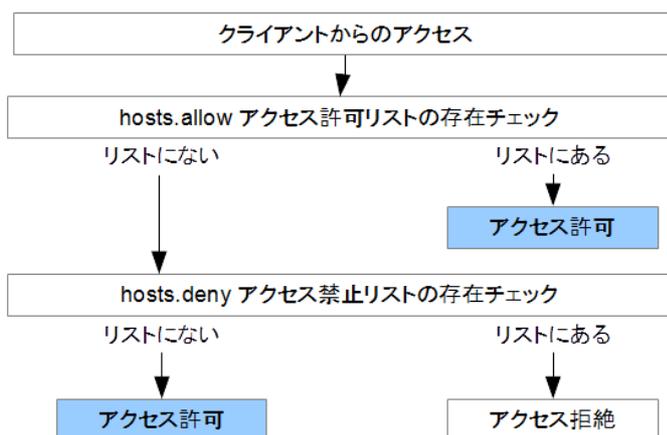
デフォルトの `hosts.allow` ファイルと `hosts.deny` ファイルには、何の設定も記述されておらず、全てのホストからの全てのサービスへの接続が許可されています。

`hosts.allow` ファイルにはアクセス許可するホストを IP アドレスまたは、FQDN によって設定します。`hosts.deny` ファイルにはアクセス拒否するホストを同様に記述します。

```
sshd: 指定するホスト 1, 指定するホスト 2 ...
```

対象が複数の場合はカンマで区切って記述します。

`hosts.allow` と `hosts.deny` ファイルによるアクセス制限の許可・拒否は以下のプロセスでチェックされます。



全てのホストを示す「ALL」を指定することもできます。これを用いれば、例えば「10.20.0.0/255.255.0.0」に属するホストを許可し、それ以外は拒否にするという設定を行う場合は、`hosts.deny` ファイルに以下を記述し

```
sshd: ALL
```

`hosts.allow` ファイルに以下を記述します。

```
sshd: 10.20.0.0/255.255.0.0
```

上記のようにネットワークアドレスとサブネットマスクを使用することに加え、「10.20.」といった表記を用いることもできます。また、ドメイン名を指定する場合は「.la.net」や「*.la.net」などが利用できます。

`hosts.allow` ファイルや `hosts.deny` ファイルの編集は、`sshd` の再起動を必要とせず、設定が反映されます。これは、クライアントからの接続要求が行われるたびにこれらのファイルを参照しているからです。これらの機能は TCP Wrapper (libwrap) により提供されています。

【練習】

1. 自ホストからの SSH サーバーへのアクセスを許可し、隣の方のホストからのアクセスを拒否するためには、どのように設定すればよいか考えます。
2. 実際に 1.の条件を満たすように、自分のホスト上の設定ファイルを編集します。
3. 隣の方のホストから自分の SSH サーバーに、ssh コマンドによるアクセスを試みてもらい、アクセスが拒否されることを確認します。
4. 設定ファイルを元に戻して、再度、隣の方のマシンからアクセスしてもらいます。
5. 正常にログインできることを確認します。

1.4 確認問題

次の確認問題を解いてください。

1. リモートログインに TELNET が推奨されず、SSH が推奨される理由を簡潔に整理します。
2. SSH を用いた通信では、サーバーの「なりすまし」攻撃からホストを守るために、ある工夫がされています。この工夫を簡単に説明してください。
3. CentOS のパッケージがアップデートされているか確認します。
アップデートパッケージがある場合はアップデートします。
4. sshd について、次の仕様が満たされているか設定の確認をします。
 - root ユーザーのログインは禁止
 - UNIX パスワードは使用許可
 - アクセスを許可するホストは、同一のネットワークセグメント内に限定
5. sshd を再起動する必要があるか否か判断し、これらの設定を有効にします。
6. useradd コマンドを用いて、新規ユーザーアカウント sshuser を作成します。パスワードを pass として設定します。
7. 自分のホストから自分のホストで動作する SSH サーバーに、ログインできるか確認します。
 - sshuser としてログインできるか否か
 - root でログインできるか否か
8. 隣のホストから自分のホストで動作する SSH サーバーにログインできるか確認してもらいます。
 - sshuser としてログインできるか否か
 - root でログインできるか否か

2 OpenSSH (2)

2.1 公開鍵暗号認証

前章の「UNIX パスワード認証を用いた SSH によるログイン」は、送信されるアカウント情報が暗号化されるため、TELNET に比べれば各段に安全性が高いといえます。しかし、暗号化されているとはいえ、アカウント情報(ユーザー名やパスワード)をネットワーク上に流しているということに変わりありません。したがって、第三者によって解読される可能性が残ります。

この章で解説する「公開鍵暗号認証」は、アカウント情報をネットワークに流さずに、確実なユーザー認証を行うことができます。

2.1.1 暗号化とは

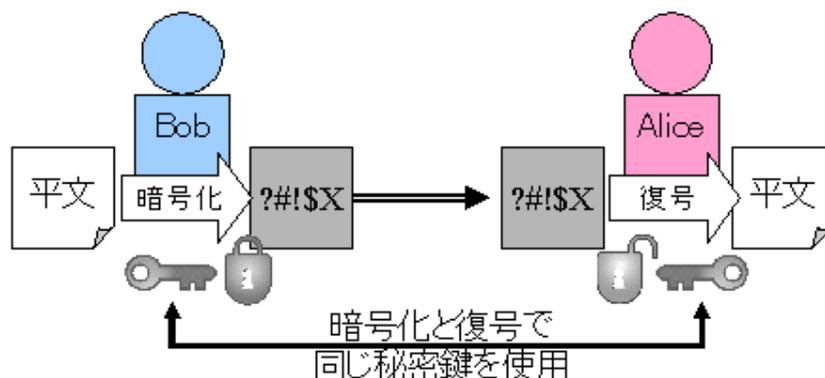
暗号化には大きく分けて2種類あります。暗号化すると二度と元に戻せない「片方向暗号(ダイジェスト)」と、鍵と呼ばれる情報を使って元に戻せる「鍵暗号」です。片方向暗号では「暗号化された情報が同じであれば、変換前のデータも同じ」という前提を用いたもので、主に認証の際に用いられます。後者は、データを暗号化したのち、再変換して元のデータに戻(復号)します。

様々な暗号化方式が存在しますが、いずれも「共通する操作」と「結果を左右する変数」が必要になります。共通する操作のことを「アルゴリズム」、結果を左右する変数を「暗号鍵」と呼んでいます。アルゴリズムがわかっても暗号鍵がなければ、解読することはできません。逆に暗号鍵を入手しても、アルゴリズムが不明では解読できません。

ただ、現代はコンピューターが極めて高性能なため、時間をかければ、いずれ解読されてしまいます。そこで、最近の暗号化技術は、より複雑なアルゴリズムを用いて、解読に膨大な時間がかかるようにすることで解読行為そのものを非現実的とするアプローチが考えられています。

2.1.2 公開鍵暗号とは

データの暗号化には、アルゴリズムと鍵が必要であることは前述の通りです。一般に暗号化のための鍵と復号のための鍵が同じ暗号を「秘密鍵暗号」と呼びます。双方で共通の鍵を使用するところから「共通鍵暗号」と呼ぶこともあります。



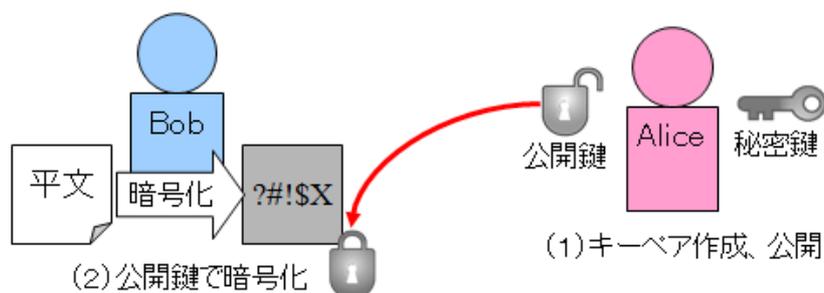
共通鍵暗号では、最も重要な「鍵」をやりとりする時点では、暗号化経路が確立していません。「鍵」をいくら暗号化しても、それを復号するための共通鍵を平文で渡すことになります。

これは「鍵配送問題」と呼ばれ、暗号の歴史の中で非常に多くの人を悩ませてきました。

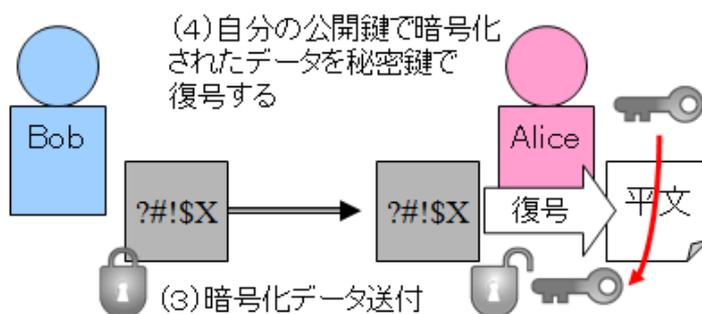
この「鍵配送問題」を解決したのが、「公開鍵暗号」です。公開鍵暗号では、「暗号化するための鍵（公開鍵）」と「復号するための鍵（秘密鍵）」の2つを uses。公開鍵はその名の通り、公開して誰でも入手できるようにし、それにより暗号化されたデータは受信者だけが持つ秘密鍵によってのみ復号できます。

以下は、BobがAliceに暗号化したメッセージを送信する手順です。

1. 受け取り側のAliceは、公開鍵暗号で必要となるキーペア（秘密鍵と公開鍵）を用意し、公開鍵を誰でもアクセスできる場所に置きます。
2. メッセージを送ろうと思うBobは、Aliceの公開鍵を入手し、それを使って自分のメッセージを暗号化します。



3. Bobは暗号化済みのメッセージをAliceへ送ります。
4. Aliceは受信したメッセージを、予め作成していた「秘密鍵」を使って復号します。



2.1.3 公開鍵暗号を用いた認証

SSH ではユーザー認証を行うために公開鍵暗号を用いることができます。公開鍵暗号を利用した認証の手続きは、以下のようなプロセスで行われます。

1. SSH クライアントでキーペア (秘密鍵と公開鍵) を作成する
2. ログイン先の SSH サーバーに、あらかじめ自分の公開鍵を登録する
3. SSH クライアントから ssh コマンドで SSH サーバーにアクセスする
4. SSH サーバーから公開鍵で暗号化されたランダムな文字列データが SSH クライアントに送信される
5. クライアントが受信したデータを秘密鍵により復号し、そのデータを SSH サーバーに送信する
6. 元の文字列とクライアントが復号して送ってきたデータが一致すれば、認証が完了する

この認証方法は、「ある公開鍵で暗号化されたデータはペアとなる秘密鍵によってのみ復号される」という性質を利用しています。

【練習】

1. 公開鍵暗号認証方式において、鍵はいくつ必要になりますか。
2. 公開鍵と秘密鍵のうち、どちらを SSH サーバー側に登録しますか。

2.2 OpenSSH における公開鍵暗号認証

2.2.1 鍵の生成

SSH サーバーで、公開鍵暗号認証を行うためには、公開鍵と秘密鍵のペアの鍵(キーペア)を作成する必要があります。この鍵はどこで作成されても構いませんが、基本的にはクライアント側ユーザーが作成します。

OpenSSH で作成できる鍵のアルゴリズムは以下の通りです。

プロトコル	SSH Ver.1	SSH Ver.2
アルゴリズム	RSA	DSA, RSA, ECDSA, ED25519

現在では、SSH1 の公開鍵認証のアルゴリズム RSA (SSH2 の RSA と名前は同じですが、互換性はありません) には脆弱性が報告されているため、SSH2 を使うようにしてください。

公開鍵と秘密鍵を作成するためには、ssh-keygen コマンドを用います。DSA というアルゴリズムを用いた暗号鍵のキーペアを作成する場合は、以下のように作成します。

```
$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_dsa):
```

鍵ファイル名の指定ができますが、特に必要ない限りデフォルトの「~/ssh/id_dsa」で構いません。エンターを押して次に進みます。

```
Enter passphrase (empty for no passphrase): [パスフレーズ入力]
Enter same passphrase again: [パスフレーズ入力確認]
Your identification has been saved in /home/student/.ssh/id_dsa.
Your public key has been saved in /home/student/.ssh/id_dsa.pub.
The key fingerprint is:03:b9:ee:61:22:0c:3d:60:11:40:79:7e:77:11:9e:93
student@h006.s142.la.net
The key's randomart image is:
+--[ DSA 1024]-----+
|++0      ..      |
| ...    ...0     |
|..0   o  E.     |
|.0 . . + ..     |
|. o . o S       |
| o . . .        |
| o . +          |
| . + .          |
| .              |
+-----+

```

続いて、パスフレーズの入力を求められます。これは暗号化された秘密鍵を復号する時に必要となるパ

スワードのようなものです。パスフレーズが設定された秘密鍵は、盗まれたとしても本人以外のユーザーには使用できません。パスフレーズは空の文字列にすることもできますが、こうしたセキュリティ向上のため複雑な文字列を登録しておきます。

ユーザーのホームディレクトリ以下のディレクトリ、ssh の中に公開鍵「id_dsa.pub」と秘密鍵「id_dsa」という、キーペアが作成されます。

```
$ ls -l .ssh
合計 12
-rw----- 1 student student 751  7月 27 20:29 id_dsa
-rw-r--r-- 1 student student 619  7月 27 20:29 id_dsa.pub
-rw-r--r-- 1 student student 171  7月 27 20:42 known_hosts
```

id_dsa (秘密鍵) の例

```
-----BEGIN DSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: AES-128-CBC, 01BFD6AF346CCE5584DE4900FA59C7E6

i2Xzp10m4epUQbltcAd0nBeK/ZTWcmGAXZLQi4P3RzIkQc6Jor4zFjz5yyZP01hh
rWVv7VsuXN+qILsWQ0ajnXFdibhQcaW/dGF8vldBIDfN2lq5y94/hhLHBjZsEreC
3p86XD8hbWkC+zz2Gtt4bxR05cz3Xwakcs5R6cVGdCxmSkkBh84EFu5Taq6fqpdM
q3yXdZZL0WZlYD801IWZfZsNYU1jiLBE1w0on3Z0AbBargfU7bL0FZKTRFvKoT3V
fhXs9SD9DlkH+AxYwu+w3456Es9f/B6LB8CQDho/xQrzUp1DdK0cXtw4kpqqII70
28eHxRUbmny1F13H/fPH0bhZXCAdo3iKTo42oDFEwDuu07zE02Fy0km/hTa1b3L
a+GwNH0vSHjV6GTVBXzyunV0Pc6kVH1WI6/Ik5Kjca4I5KIEoae4ZtAUr10di0Yo
L+U0u9jbL3yULipQsvQgFb1aL+1jEW0uRgZ20Ie3UThCjr3a1ieuDvZIfp3wgi+
0uA7Jfbro0EEtNkInMpJ6Zu0Mcw3Lrk13qbIAG+OXkngc2kMkeQCdTh2Unv5yRGR
gUoz1mcfwVGgW+2WfqdHRg==
-----END DSA PRIVATE KEY-----
```

id_dsa.pub (公開鍵) の例

```
ssh-dss
AAAAB3NzaC1kc3MAAACBALn2Gc7zddXGzRvxQiX07HtNTjC/oWiLUUz+hQ0mMo
Gnzo4KQsdmQHW0Jaayf6UcM1tPeC+G1nQ6Vwb9LDQCh2FavBJGR0jP+GM6Dq42rpzH5c
mNNLN4/XfRcTx7039bqXRh8K7rXhjBnkHhWG+hcZ+6PBCdMJVE2axgu/8gP+fpAAAAFQ
Cw2ZveluorBL5GIhLrXIn2dqXLqQAAAIb12cMbaO9GpN6ETohFmAWMM01AbnV3YkUsr
IkIVhC0wvBZb0UCmdxazhdQo4uB8KgvnwNqQ5UdiZXkv61XexK92R0cDsNUs3M4GKGIE
P4L1h8J6N5Cfv/EkrPwSImDQLeGhxSDJQ44HCG6D0Z6xc8HeAbLEFCTiuTFAvbkz2QCw
AAAIEAiU5UYy7sthZdMcuEjQUzRE7kM/QKpGIJQv0l6UbIJ5FmaOZI6Apc/o+bXeZndS
PiW9iNsEeG7Th/H3X3SajuEV2PKtipnEB6gBemlJsmITD+bbEJCHEvyICPwCpEoS1x5Q
t00lIFEI+0mbjbZENlrfMc50v95/ms9igZzf0Vm+E= student@h006.s142.la.net
```

アルゴリズム DSA を用いた場合の秘密鍵・公開鍵ファイルは上のようなテキストデータになっています。

アルゴリズム RSA を用いる場合もこれと同様で、鍵生成の際のアルゴリズムとして、「-t rsa」とオプション指定します。

生成されるアルゴリズムと鍵ファイルは次の通りですが、SSH1 は脆弱性があるため特別な理由がない限り使用しません。

プロトコル	アルゴリズム	秘密鍵	公開鍵
SSH1	RSA	identity	identity.pub
SSH2	DSA RSA ECSDA ED25519	id_dsa id_rsa id_ecdsa id_ed25519	id_dsa.pub id_rsa.pub id_ecdsa.pub id_ed25519.pub

2.2.2 SSH サーバーへの公開鍵の登録

公開鍵認証を行うには、ログイン先の SSH サーバーに、生成した公開鍵を予め登録する必要があります。特に管理者の設定など極めて重要な公開鍵ファイルを登録するには、あえて CD-ROM や USB メモリなどオフライン媒体を用いることもあります。

ログイン先のユーザーのホームディレクトリ以下の `.ssh/` の中の `authorized_keys` というファイルに公開鍵ファイルの内容を書き込むことで、公開鍵の登録が完了します。

```
$ cat ./id_dsa.pub >> ~/.ssh/authorized_keys
```

上記のようにコマンドを入力することで、`authorized_keys` ファイルがない場合は新たに作成され、既にある場合は内容が追記されます。

特に指定しない限り `authorized_keys` ファイルを上記のように作成した場合、パーミッションは「664」となります。グループ内の他のユーザーに変更されないようにするために少なくとも「644」または、セキュリティ対策として「600」に変更する必要があります。また、`authorized_keys` ファイルがあるディレクトリ (`~/.ssh`) のパーミッションも「700」に設定しておく必要があります。

2.2.3 公開鍵認証を用いたログイン

SSH サーバーに公開鍵を登録したら、ログイン準備の完了です。ログインのためには、前回と同様に `ssh` コマンドを用います。

```
$ ssh student@10.20.142.6
Enter passphrase for key '/home/student/.ssh/id_dsa': パスフレーズの入力
Last login: Thu May 7 13:07:36 2017
```

公開鍵の登録とパーミッションの設定が適切ならば、秘密鍵を使うためのパスフレーズ入力が求められます。デフォルトでは、秘密鍵ファイルとして、`~/.ssh/` ディレクトリ内の `id_dsa` や `id_rsa` などアルゴリズムに応じたファイルが自動的に採用されます。

別の秘密鍵を指定したい場合は、オプション「`-i` 鍵ファイル名」を使用します。

```
$ ssh -i /home/student/.ssh/dsakey student@10.20.142.6
```

上の例では、「`dsakey`」というファイル名を秘密鍵に指定しています。

[練習]

自分のマシンを SSH サーバーとし、公開鍵認証を利用したログインを行います。

1. ユーザー `student` になって、アルゴリズム DSA の認証鍵のキーペアを作成します。認証鍵の名前はデフォルトのまま構いません。
2. 公開鍵を自分のホストに登録します。例にならって、`id_dsa.pub` ファイルの内容を `authorized_keys` ファイルに追記します。
3. `authorized_keys` ファイルにふさわしいパーミッションに変更します。
4. `ssh` コマンドを用いて、自分のホストにユーザー `student` としてログインします。

2.2.4 SSH サーバー側の設定

インストールされた SSH サーバーはデフォルトで公開鍵認証方式を許可する設定になっています。これらの設定も SSH サーバーの設定ファイルである、「`/etc/ssh/sshd_config`」ファイルにて行います。

公開鍵認証を許可するための設定項目

設定項目	解説 (規定値)
<code>RSAAuthentication</code>	SSH1 の RSA 認証 (アルゴリズムとして RSA を用いた公開鍵認証) を許可する (yes)。
<code>PubkeyAuthentication</code>	SSH2 の公開鍵認証を許可する (yes)。
<code>AuthorizedKeysFile</code>	SSH サーバーが参照するユーザーの公開鍵ファイルをホームディレクトリからの相対パスで指定する (<code>.ssh/authorized_keys</code>)。

2.3 安全なファイル転送

2.3.1 scp コマンド

scp コマンドは、暗号化された経路を用いて、ローカルホストとリモートホストの間でファイルのやりとりを行うためのコマンドです。cp コマンドのように使えますが、ファイル名の指定に、リモートホストのファイルを指定することができます。

```
scp コピー元のファイル コピー先のファイル
```

ファイルは以下の形式で指定します。

```
ユーザー名@ホスト名:ファイル名
```

ファイル名は指定したユーザーのホームディレクトリを起点とします。相対パス名を使う場合には注意が必要です。

例: ホスト 10.20.142.6 のファイル /etc/hosts をカレントディレクトリ(.)にコピーする。

```
$ scp student@10.20.142.6:/etc/hosts .
```

scp コマンドを用いるときにユーザー認証が行われますが、これは ssh コマンドの時と同様の認証が用いられます。

【練習】

scp コマンドを用いて、隣のホストから自分のホストにファイル転送を行います。この時 UNIX パスワード認証が行われるため、事前に UNIX パスワード認証が許可されているかどうか確認してください。

1. 隣のホストにあるファイル /home/student/.bashrc を、ホームディレクトリの中に bash.cp としてコピーする。
例) `scp student@お隣:/home/student/.bashrc ~/bash.cp`
2. UNIX パスワードが求められるのでこれを入力します。
3. ファイル転送が正常に行われることを確認します。

2.3.2 sftp コマンド

sftp も scp 同様に OpenSSH 付属のツールです。sftp は対話的にファイル転送が行えます。

```
sftp ユーザー名@ホスト名
sftp> サブコマンド
:
```

sftp サーバーは SSH サーバーの付属の機能(サブシステム)として、実行することができます。
/etc/ssh/sshd_config ファイルに以下のような記述があると(デフォルトで設定済み)、sshd 起動時に sftp サーバー「/usr/libexec/openssh/sftp-server」も実行されます。

```
# override default of no subsystems
Subsystem          sftp          /usr/libexec/openssh/sftp-server
```

クライアントから sftp サーバーに接続するためには、sftp コマンドを用います。sftp コマンドは ftp コマンドと同様にログイン終了後、get や put サブコマンドを利用することにより、ファイルのやりとりを行います。

主な sftp サブコマンド

sftp サブコマンド	解説
put ローカルファイル [リモートファイル]	ファイルのアップロード。第二引数を省略すると、ローカルファイルと同じ名称が採用される。
get リモートファイル [ローカルファイル]	ファイルのダウンロード。第二引数を省略すると、リモートファイルと同じ名称が採用される。
cd リモートディレクトリ	サーバー側のディレクトリ移動
lcd ローカルディレクトリ	クライアント側のディレクトリ移動
pwd	サーバー側のカレントディレクトリ表示
lpwd	クライアント側のカレントディレクトリ表示
ls	サーバー側のファイル一覧表示
quit	sftp の終了
help	サブコマンドの簡単な使い方表示

[練習]

1. sshd の設定ファイルを参照し、sftp サーバーが利用できることを確認します。
2. 必要に応じて、sshd を再起動します。
3. 自分のホストに対して、sftp コマンドを用いてアクセスします。
4. ファイルのアップロード、ダウンロードのテストを行います。

2.4 確認問題

1. 公開鍵暗号を用いた認証では UNIX パスワード認証に比べて、どのような点で安全性が優れていますか。
2. 公開鍵を用いた認証を行う操作手順(準備)を説明してください。
3. 以下の演習で隣の方に利用してもらうユーザーアカウントを作成します(ユーザー名「sshguest」、パスワード「guest」)。
4. 自分のホストで student ユーザーにて、認証鍵(SSH2・RSA)のキーペアを作成します。
5. 作成した認証鍵のキーペアのうち、公開鍵を相手のサーバーにコピーします。安全なファイル転送の手法である sftp、または scp を利用して、自分の公開鍵を隣のユーザー「sshguest」のホームディレクトリにコピーまたはアップロードします(scp、sftp どちらを使うかは講師の指示に従って下さい)。
6. 一旦 UNIX パスワード認証で、隣の sshguest にログインします。
7. ホームディレクトリにある student の公開鍵を登録します。.ssh ディレクトリがなければ作成します。このとき、セキュリティ対策として、.ssh ディレクトリのパーミッションを「700」に変更する必要があります。この.ssh ディレクトリの中に authorized_keys ファイルとして鍵登録を行います。
8. パーミッションの確認を行い、適切なパーミッションに調整します。
9. 一旦ログアウトします。
10. 再度、隣の sshguest としてリモートログインし、パスフレーズが求められ、公開鍵暗号認証方式でログインできるか確認します。

パスフレーズではなくパスワードの入力が求められる場合は以下のことを確認します。

- 設定ファイルや鍵ファイルの所有者やパーミッションは適切か
- 接続先のホストで sshd が起動しているか
- sshd の設定が適切なものになっているか
- アクセス制限が適切なものになっているか

3 メールサーバー (1)

3.1 メール配信の仕組み

電子メール(以降、メールとします)の送受信には、メールサーバーが利用されます。普段、メールの送受信に利用しているメーラー(メールクライアント)はメールサーバーにアクセスすることで、メールの送信要求や、受信要求を行います。

メールの配送の仕組みは一見複雑に見えますが、実は郵便の配達方法を元にしています。メールサーバーの構築にあたっては、メール配信の仕組みを理解していることが必須です。ここでは、メールクライアントからメールが送信されて、宛先のユーザーがそのメールを読むまでの流れを見ていくことにしましょう。

3.1.1 メールの作成と送信

通常、メールは Outlook や Thunderbird などに代表されるメールクライアントソフト(通称メーラー)を利用して、メールを作成します。メールには相手に伝えたい内容を書きますが、加えて、宛先(受け取り手)のメールアドレス、送信元(自分)のメールアドレス、件名なども記述します。

メールを書き終わると、メールクライアントで「送信」を行います。ここでいう送信とは、メールクライアントの設定であらかじめ登録してあるメールサーバー(契約しているプロバイダのメールサーバーなど)に、メールの送信を依頼することを指します。

こうして送信されたメールは、最終的には宛先のメールサーバーに送り届けなければならないのですが、直接相手のメールサーバーに届けることはせず、登録されている手近なメールサーバーに送信を依頼することになっています。みなさんが手紙を書いた後、自分で相手の近くの郵便局まで足を運ぶわけではなく、近くの郵便ポストに投函し、配達には郵便局に任せてしまうのと同じと考えればよいでしょう。

メールクライアントからメールサーバーにメールを受け渡すときには、SMTP(Simple Mail Transfer Protocol)が利用されます。SMTP は 1982 年に RFC821 として制定された後、2001 年に RFC2821、2008 年に RFC5321 と段階を経て改訂されています。

3.1.2 メールサーバーでの配送

メールクライアントから送信要求を受け付けたメールサーバーは、本来届けられるべきメールサーバーにメールを配送しようとしています。

まず始めに、どのメールサーバーにメールを送り届けるべきかを調べます。宛先メールアドレスのドメイン部(@以降の部分)について、DNS サーバーに MX レコードの問い合わせを行います。配送すべきメールサーバーの IP アドレスがわかったら、SMTP を利用してそのメールサーバーにメールを送り届けます。

これで、宛先に対応するメールボックス(メールの最終的な配送先)があるメールサーバーにメールが届けられました。メールを受け取ったメールサーバーは、メールボックスにメールを格納します。

3.1.3 メール受信

宛先のメールボックスにメールが配送されても、まだメールの受け取り手がメールを目にしたわけではありません。郵便でいえば、郵便局の私書箱に手紙が届けられている状態です。受け取り手にメールを開封してもらう必要があります。

メールボックスに配送されたメールを参照するにはいくつかの方法がありますが、もっとも一般的なのが、クライアント側にメールをダウンロードさせる方法です。そのためのプロトコルとして、POP (Post Office Protocol) がよく利用されます。現在主に使用されているのは Version3 (POP3) です。

こうして、POP によりメールクライアントにダウンロードされたメールは宛先 (受信者) に読まれることになるわけです。

POP は RFC918 から始まり、1000 番代 2000 番代などを多数経て、2007 年の RFC5034 に至るまで改訂され続けています。

最近ではメールをダウンロードせずに、直接メールサーバーのフォルダを操作する IMAP (Internet Message Access Protocol, 現在は Version 4 が主流) も多く用いられています。IMAP では受信したメールはサーバー上に保存されますが、POP ではクライアントにダウンロードされるため原則としてサーバーには残りません。

IMAP はサーバー側で処理を行うため、サーバーの性能やディスク容量を多く消費するといったデメリットはありますが、メールがサーバーで一元管理されているため複数のクライアントからアクセス可能、セキュリティの向上や情報漏えいに強いといったメリットもあります。

IMAP は最初 RFC1730、1731 が規定されたのち、5957、6154、6203、6237 など機能追加が続いています。

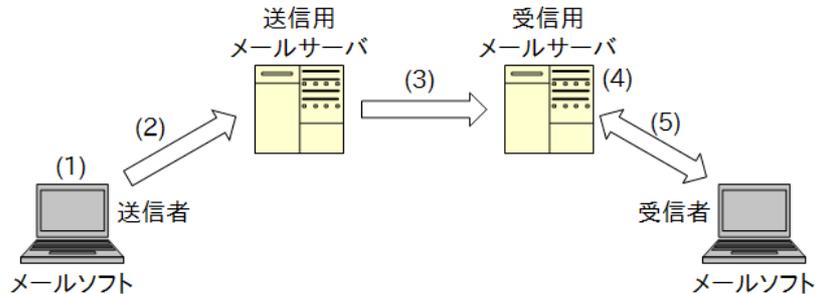
[練習]

インターネット検索エンジンを使って、上記 RFC のいくつかを調べてください。

3.2 メールサーバーの役割

3.2.1 ローカル配送とリモート配送

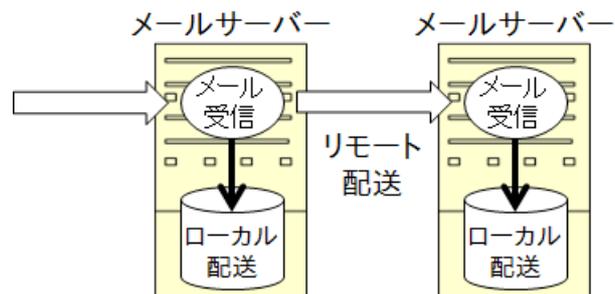
メールの配送の仕組みを整理すると次のようになっています。



1. メールクライアント(メーラー)でメールを作成する
2. メールクライアントが送信用メールサーバーにメールを送る
3. 送信用メールサーバーが、宛先のメールサーバーを調べ、そのメールサーバーへメールを送る
4. メールを受け取ったメールサーバーが、宛先のメールボックスにメールを配置する
5. 宛先ユーザーがメールサーバーからメールをダウンロードし、メールを読む

メールサーバーの動作に注目し、どのように処理を行うかを見てみましょう。メールサーバーの機能を一言で言ってしまうと、メールクライアントや他のメールサーバーから受け取ったメールを適切に配送することです。

配送は大きく2つのケースがあります。1つはサーバー内にメールボックスがあり、そのメールボックスにメールを格納する「ローカル配送」。もう1つは他のメールサーバーにメールボックスがあるため、そのメールサーバーに転送する「リモート配送」です。

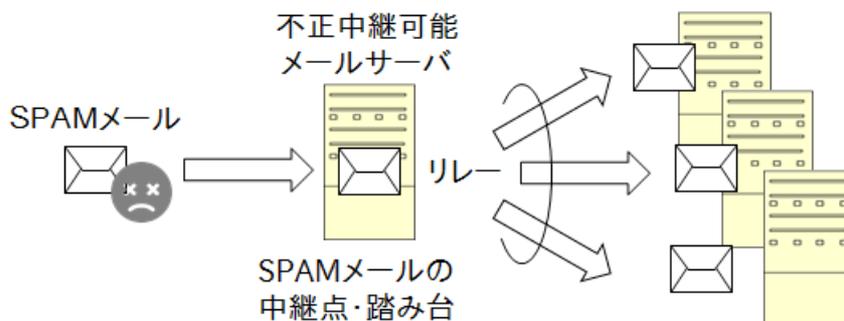


3.2.2 メールの中継と制限

リモート配送の場合、受け取ったメールはそのまま他のメールサーバーに転送されます。このように自分宛ではないメールを受け取り、リモート配送することを「メールの中継(リレー)」といいます。従来はメールの中継は特に制限を設けず、自分宛でないメールでも受け取り、本来の宛先サーバーへ配送することで、円滑にメールの配送を実現していました。

現在では迷惑メール(SPAM)が社会的な問題となり、不必要にメール中継を許すことは望ましくない行

為となりました。迷惑メールの送信者は、中継してくれるメールサーバーを見つけ、(迷惑)メールの中継を依頼します。これにより、送信者は自身の素性を隠して、迷惑メールの送信が可能となります。このように誰にでもリレー(オープンリレー)すると、迷惑メールのほう助となるだけでなく、サーバーのリソースを浪費することになります。したがって、不必要に中継をしないように設定することが必要不可欠なのです。



しかし、「メールの中継が不要」とは言い切れません。メールクライアントは、手近なメールサーバーに配送を依頼する時に中継が必要です。

通常は、会社や組織のメールクライアントからの中継を受け、それ以外は自分宛のメールのみを受け取るように設定します。

メールサーバーは受け取ったメールに対して以下のいずれかの処理を行います。

1. 宛先のユーザーが自分のホスト内に存在すれば、メールボックスにその内容を保存する(ローカル配送)。
2. 宛先のユーザーが自分のホスト内に存在しなければ、配送されるべきメールサーバーを DNS から判断し、そのメールサーバーへメールを中継する(リモート配送)。
3. 宛先のユーザーが存在しないなどの送信エラーが発生すれば、送信者にその旨を通知する。

このメールサーバー間の通信にも前述の SMTP が用いられます。これらの手順を繰り返すことで、目的のメールサーバーにメールが配送され、宛先に対応するメールボックスに格納されます。

3.2.3 宛先と名前解決

メールアドレスには「student@s142.la.net」という表記をういます。これは「ユーザー名@ドメイン名」または「ユーザー名@ホスト名」です。この表記でメールが届くためには、名前解決が適切に行われている必要があります。

メールサーバーはメールを受け取ると、宛先に記されているドメイン名またはホスト名を見て、該当するゾーンのネームサーバーに問い合わせをします。このとき参照されるのは、ゾーンの「MX レコード (mail exchanger)」です。MX レコードとして記されたホストは、そのゾーン内のメールサーバーを表します。

例: info@linuxacademy.ne.jp のメールサーバーを調べる

```
$ nslookup -query=mx linuxacademy.ne.jp
Server:          10.20.250.1
Address:         10.20.250.1#53

Non-authoritative answer:
linuxacademy.ne.jp mail exchanger = 1 ASPMX.L.GOOGLE.COM.
...
```

この例では、linuxacademy.ne.jp ドメイン宛てのメールは ASPMX.L.GOOGLE.COM. というサーバーが受け取ることを示しています。

[練習]

自身のメールアドレスを元に、MX レコードを調べてください。

3.2.4 メールの受信

メールボックスに格納されたメールを読むには、メールクライアント使います。

例えば、student@example.net から teacher@la.net に対して送られたメールの場合、以下のようになります。

- teacher は、メールクライアントを使用して受信用メールサーバーに着信を問い合わせます (このとき、ユーザー認証=本人確認を行います)。
- 届いていたメールをダウンロードします。
- メールの内容を確認し、削除・返信等を行います。
- 返信・新規メールは、送信メールサーバーへ SMTP を使って送信します。

受信用メールサーバーは、POP サーバーや IMAP サーバーが務めます。多くの場合、送信用メールサーバーと受信用メールサーバーは同じホストで動作しており、総じて「メールサーバー」と呼ばれます。

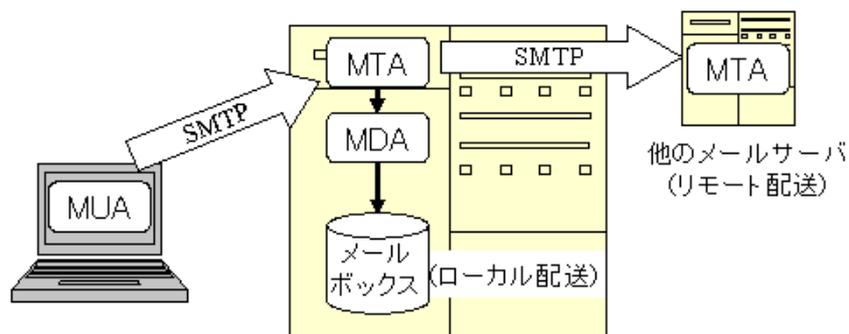
メールクライアントは SMTP/POP/IMAP といったサーバーとのやりとり以外に、アドレス帳や定型文書

(テンプレート)といった機能を持つものが多く存在します。メールサーバーの構成要素

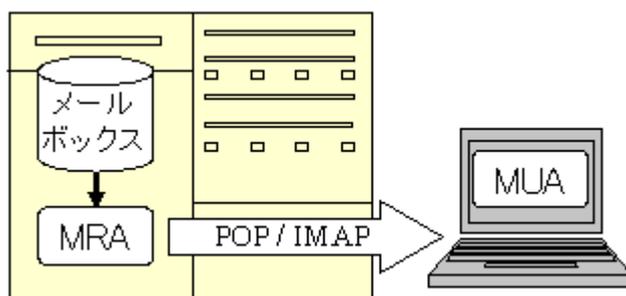
ここまで解説したメールに関わる機能を整理すると以下のようになります。

機能名	解説
MUA (Mail User Agent)	いわゆるメーラ。メールの送受信およびアドレス帳などの管理。
MTA (Mail Transfer Agent)	SMTP によってメールを受取り、自身へローカル配送または、他のメールサーバーへ転送を行う。
MDA (Mail Delivery Agent)	MTA からローカル配送分を受け取り、各ユーザーへ割り振り。最近ではその際にスパムやウイルスチェックを行う。
MRA (Mail Retrieval Agent)	ユーザー認証を行った後、ローカル(メールボックス)の内容をユーザーへ提供。POP は単純なダウンロード、IMAPはディレクトリ操作を行う。

送信の流れ



受信の流れ



3.3 メールの構造

メールの構造はまず、エンベロープとメール本体に分けられます。メールクライアント (MUA) でメールを確認する場合はメール本体のみが参照できます。

3.3.1 エンベロープとメール本体

エンベロープ (envelope) は「封筒」という意味があり、メール配送に必要な情報が記載されます。特にエンベロープ中の「RCPT TO」という項目は、メールの実際の配送先が示されており、最も重要な情報になります。

メール本体はヘッダとメール本文という構成になっていて、通常 MUA によって表示されるのはヘッダの一部と本文のみです。多くの MUA は送信者・宛先・件名といった一部のヘッダ情報のみを表示しています。

	エンベロープアドレス	ヘッダの To、From フィールド
説明	メールの配信に利用	メールの内容の一部で、配送には使用されない
手紙の場合	封筒に書かれた宛名 郵便局はこれを見て配達する	便箋の中に書かれた名前

3.3.2 ヘッダとメール本文

前述のようにメール本体も2つの部分に分かれています。最初にヘッダと呼ばれる情報があり、その後に本文が続きます。ヘッダにはメールに関する付加情報 (宛先アドレス・送信元アドレス・件名等) が、一定のルールに基づいて記述されます。ヘッダと本文の間は一行の空行によって区切られています。

ヘッダと本文の例

```
From student@h231.s142.la.net  Fri May  8 11:44:19 2017
Return-Path: <student@h231.s142.la.net>
Date: Fri, 8 May 2017 11:44:19 +0900
From: student <student@h231.s142.la.net>
Message-Id: <201705080244.n482iJiQ028392@h231.s142.la.net>
To: teacher@h006.s142.la.net
Subject: Question about Mail System
```

先生

受講生の〇〇です。メールについて質問です。

メールサーバーが配送先として利用しているアドレスは、ヘッダの To アドレスではなく、エンベロープで指定された RCPT TO の値と理解してよろしいでしょうか。

To アドレスはいわば封書の中の便せんに記載された「XX 様」と同様の役割しか果たしていないということなのですね。

ヘッダに記載されている情報の1つ1つをフィールド(field)と呼びます。メールヘッダについては RFC821 と 2076 で規定されていますが、代表的なフィールドとして以下のようなものが挙げられます。

代表的なヘッダフィールド

フィールド	解説
From	送信者のメールアドレス
To	宛先のメールアドレス
Delivered-To	実際に配送されたメールアドレス エンベロープで指定された「RCPT TO:」の値が格納される
Date	メールの送信日時
Subject	メールの件名
Message-ID	メールを識別するために利用される ID インターネット上で一意になるよう考慮されている

[練習]

メール本文を印刷した書面だけでは、証拠として採用されない理由を考えてください。

参考:「堀江メール問題」

3.4 電子メールのプロトコル

3.4.1 SMTP

メールクライアントやメールサーバーなどがメールを送信する場合に利用されるのが SMTP (Simple Mail Transfer Protocol) です。SMTP では、最初に自分のホスト名を名乗り (EHLO)、その後エンベロープとして、送信元アドレス (MAIL FROM)、宛先アドレス (RCPT TO) に基づき、メールの本体 (DATA) を送信します。

例: student 宛にメールを送る (太字がリクエスト)

```
220 h231.s142.la.net ESMTP Postfix
EHLO h100.s142.la.net
250-h100.s142.la.net
    (中略)
250 DSN
MAIL FROM: root@h100.s142.la.net
250 2.1.0 OK
RCPT TO: student@h231.s142.la.net
250 2.1.5 OK
DATA
354 Enter data with <CR><LF>.<CR><LF>
Date: Fri, 8 May 2017 11:44:19 +0900
From: root <root@h100.s142.la.net>
Message-Id: <201705080244.n482iJiQ028392@h100.s142.la.net>
To: student@h231.s142.la.net
Subject: This is Test Mail

テストメールです!
.
250 2.0.0 OK: queued as E8188806F6D
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

主な SMTP コマンド

コマンド	解説
EHLO ドメイン名	SMTP 通信の開始を表す
MAIL FROM:メールアドレス	送信者のメールアドレスを通知する
RCPT TO:メールアドレス	宛先のメールアドレスを通知する
DATA	メール本体の始まりを表わす 「.」だけの行の入力によって入力終了となる
QUIT	通信の終了

「MAIL FROM」や「RCPT TO」ではエンベロープ情報を送信しており、「DATA」以降の部分はメールの本体となっています。

3.4.2 POP

メールクライアントからメールをダウンロード(受信)するときに POP3 (Post Office Protocol Ver.3) が利用されます。POP3 では、メールボックス内に格納されているメールの一覧の取得や、各メールデータを取得することができます。

例: ユーザー student のメールボックスへのアクセス

```
+OK Dovecot ready.
USER student
+OK
PASS himitu
+OK Logged in.
LIST
+OK 2 messages:
1 587
2 3671
.
UIDL
+OK
1 000000014c03ac48
2 000000014a04bd59
.
RETR 1
+OK 587 octets
(メール本体:省略)
.
DELE 1
+OK Marked to be deleted.
QUIT
+OK Logging out, messages deleted.
Connection closed by foreign host.
```

主な POP コマンド

コマンド	解説
------	----

USER [ユーザー名]	ユーザー認証のためのユーザー名
PASS [パスワード]	USER で指定したユーザーに対応するパスワード
STAT	メールボックスに格納されているメールの件数とそのバイト数
LIST	各メールのサイズを取得する
UIDL	各メールに固有の ID を取得
RETR	指定したメール本体を取得する
DELE	指定したメールをメールサーバーから削除する
QUIT	メールサーバーとの接続を切断する

[練習]

Gmail、Yahoo!メールなどはパソコンに導入したメールソフト (Outlook、Thunderbird など) と同期が可能です。受信プロトコルは POP3 と IMAP が用意されています。どちらを使うべきか考えてください。

4 メールサーバー (2)

4.1 Postfix

4.1.1 Postfix の概要

メールサーバーには sendmail、Postfix、qmail、exim などいくつかのソフトウェアが存在していますが、この章では多くの UNIX 系 OS で採用されている Postfix をとりあげます。

Postfix は IBM トーマス・J・ワトソン・リサーチ・センターの Wietse.Z.Venema 氏が開発したメールサーバーソフトで、「高速で、設定が容易、願わくは安全に」動作することを目標としています。

Postfix をはじめとする MTA (Mail Transfer Agent) は、ポート 25/TCP を LISTEN して接続を待ち受け、メールクライアントや他のメールサーバーから SMTP によって送られてきたメールを処理します。自身のメールボックスにメールを格納、または必要に応じて他のメールサーバーに配送 (中継・リレー) します。

4.1.2 Postfix の設定

CentOS では、Postfix が標準メールサーバーとして採用されているため、特にインストールする必要はありません。インストールされていない場合は、yum コマンドでインストールしてください。

```
# yum install postfix
```

Postfix の設定ファイル /etc/postfix/main.cf は、「設定項目 = 設定値」の形式で記述されます。設定値が複数ある場合には「,」で区切ります。また「#」を行頭に記述することで、その行がコメントアウトされます。

設定ファイル main.cf の例 (抜粋)

```
# Global Postfix configuration file. This file lists only a subset
# of all parameters. For the syntax, and for a complete parameter
# list, see the postconf(5) manual page (command: "man 5 postconf").
...
queue_directory = /var/spool/postfix
...
command_directory = /usr/sbin
...
daemon_directory = /usr/libexec/postfix
...
```

設定値として、既に設定している他の設定項目の値をそのまま利用したい場合は、「\$設定項目名」を設定値として指定します。例えば、設定項目名 myhostname に設定した値をそのまま利用したい場合には、その値として「\$myhostname」と指定します。

【練習】

main.cf を less コマンドなどで参照し、全体の構造を把握します。

主な設定項目としては以下のものがあります。

設定項目	解説
myhostname	メールサーバーのホスト名を指定する。
mydomain	メールサーバーが所属するドメインのドメイン名を指定する。
myorigin	メールアドレスの@以下が指定されていない場合に補完される値。通常は\$myhostname か\$mydomain を指定する。
inet_interfaces	SMTP での接続を待ち受けるネットワークインターフェイス (IP アドレス) を指定する。all 指定により全てのインターフェイスから待ち受け可能になる。
mydestination	ローカル配送を行うメールアドレスのホスト部。宛先メールアドレスのホスト部がこの値に含まれているとローカル配送する。
mynetworks	中継 (リレー) が必要なメール送信要求を受付けるホスト (ネットワーク) を指定する。

全ての設定項目について設定値を確認したい場合は、postconf コマンドを利用します。postconf コマンドは現在の main.cf を解析した上で、Postfix が動作する場合に適応される設定値の一覧を出力します。

「-n」オプションを指定すると、デフォルトの値と異なる値が設定されている設定項目のみ出力します。

```
# postconf -n
alias_database = hash:/etc/aliases
alias_maps = hash:/etc/aliases
command_directory = /usr/sbin
config_directory = /etc/postfix
daemon_directory = /usr/libexec/postfix
data_directory = /var/lib/postfix
...
```

[練習]

main.cfを編集し、次の値を設定します。ホスト名、ドメイン名、IPアドレスは環境に応じて変更します。

```
myhostname = hXXX.s142.la.net
mydomain = s142.la.net
myorigin = $myhostname
inet_interfaces = all
mydestination = $myhostname
mynetworks = 10.20.xxx.yyy, 127.0.0.1, localhost
```

postconfコマンドに「-n」オプションを指定して、1.で設定した内容を確認します。

これにより、全てのネットワークインターフェイスについてSMTPでの接続を待ち受け、宛先アドレスのホスト部が「h006.s142.la.net」になっているメールをローカル配送することになります。また、10.20.142.6 および 127.0.0.1 (localhost) から接続された場合については、リモート配送を(中継が必要になるメールも)受け付けることになります。

4.2 Postfix の利用

4.2.1 Postfix の起動と停止

Postfix を起動するには、ユーザー root になり、systemctl コマンドを利用します。引数には、start や stop の他に、再起動させる restart、稼働状況を確認する status などもあります。

Postfix の起動

```
# systemctl start postfix
```

Postfix の停止

```
# systemctl stop postfix
```

Postfix が起動すると「master」「qmgr」「pickup」という3つのプロセスが常駐します。この中でも、master というプロセスは Postfix 全体を制御しており、他の関連するプロセスを起動したり停止したりします。Postfix に関連するプロセスはプロセス master の子プロセスとなります。

```
# systemctl status postfix
● postfix.service - Postfix Mail Transport Agent
   Loaded: loaded (/usr/lib/systemd/system/postfix.service; enabled; vendor preset: disabled)
   Active: active (running) since 火 2017-07-28 08:25:36 JST; 6h ago
     Process: 7096 ExecStop=/usr/sbin/postfix stop (code=exited, status=0/SUCCESS)
     Process: 7113 ExecStart=/usr/sbin/postfix start (code=exited, status=0/SUCCESS)
     Process: 7111 ExecStartPre=/usr/libexec/postfix/chroot-update (code=exited, status=0/SUCCESS)
     Process: 7108 ExecStartPre=/usr/libexec/postfix/aliasesdb (code=exited, status=0/SUCCESS)
    Main PID: 7185 (master)
      CGroup: /system.slice/postfix.service
             └─7185 /usr/libexec/postfix/master -w
                └─7186 pickup -l -t unix -u
                   └─7187 qmgr -l -t unix -u

7月 28 08:25:36 h006.s142.la.net systemd[1]: Starting Postfix Mail Tra...
7月 28 08:25:36 h006.s142.la.net postfix/master[1250]: daemon started ...
7月 28 08:25:36 h006.s142.la.net systemd[1]: Started Postfix Mail Tran...
Hint: Some lines were ellipsized, use -l to show in full.
```

【練習】

1. Postfix を起動します。
2. ps コマンドを利用して、Postfix に関連するプロセスが起動していることを確認します。また、各プロセスについて実行ユーザー権限がどのようになっているかを確認します。
3. pstree コマンドを利用して、Postfix に関連するプロセスの親子関係を確認します。

4.2.2 メールの配送とメールボックス

Postfix がメールを受信するには、2つのケースが考えられます。1つは、ネットワークを通じて、メールクライアントや別のメールサーバーから送信要求を受け付ける場合です。この場合、SMTP を利用して通信が行われます。2つめは、サーバー内から mail コマンドなどのメーラー (MUA) を通じて送信する場合です。

mail コマンドを利用して送信する場合は、引数に宛先のメールアドレスを指定して実行。その後、Subject を入力してからメールの本体を入力します。メール本体の入力が終わったら、行頭で「.」を入力しエンターキーを押します。

```
# mail student@h006.s141.la.net
Subject: Test mail
This is test mail
.
EOT
```

この後、Postfix は配送処理を行うこととなります。今回の場合は、ローカル配送を行うことになるため、メールボックス (/var/spool/mail/ユーザー名) にメールが格納されます。

```
# cat /var/spool/mail/student
From root@h006.s142.la.net Tue Jul 28 15:07:39 2017
Return-Path: <root@h006.s142.la.net>
X-Original-To: student@h006.s142.la.net
Delivered-To: student@h006.s142.la.net
Received: by h006.s142.la.net (Postfix, from userid 0)
        id D97BDBA; Tue, 28 Jul 2017 15:07:39 +0900 (JST)
Date: Tue, 28 Jul 2017 15:07:39 +0900
To: student@h006.s142.la.net
Subject: Test mail
User-Agent: Heirloom mailx 12.5 7/5/10
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Message-Id: <20150728060739.D97BDBA@h006.s142.la.net>
From: root@h006.s142.la.net (root)

This is test mail
```

【練習】

1. 上記の例を参考にして、mail コマンドを利用して、ユーザー student 宛にメールを送信します。
2. ユーザー student のメールボックスに、1.で送信したメールが格納されていることを確認します。
3. telnet でローカルホストの 25 番ポートに接続して、SMTP を利用して、メールを送信します。入力するリクエストについては「SMTP」(p.39) 参照のこと。

```
# telnet localhost 25
Trying ::1...
Connected to localhost.
Escape character is '^]'.
220 h006.s142.la.net ESMTP Postfix
EHLO h006.s142.la.net
250-h006.s142.la.net
      (省略)
250 DSN
MAIL FROM: root@h006.s142.la.net
250 2.1.0 Ok
RCPT TO: student@h006.s142.la.net
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: test mail

test mail
.
250 2.0.0 Ok: queued as 4C76C99CF2
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

4. 次の配送を試みます。うまくいく場合、うまくいかない場合について理由を考えます。
 - a) telnet でローカルホストに接続し、隣のマシンのユーザー student 宛にメールを送信します。
 - b) telnet で隣の方のホストに接続し、隣のマシンのユーザー student 宛にメールを送信します。
 - c) telnet で隣の方のホストに接続し、自分のホストのユーザー student 宛にメールを送信します。

リモート配送(中継・リレー)の要求は、設定項目 mynetworks で設定したホストやネットワークに限定されます。許可されない中継は、SMTP コマンド「RCPT TO」を指定した時点で、送信要求が拒否されます。

4.2.3 Postfix のログ

Postfix のログは、システムログを通じて /var/log/maillog に格納されます。いつどのようなメール送信要求を受け付け、どのように配送したかを把握することができます。

配送にあたって様々な情報が記録されますが、各メールには固有の ID が割り当てられており、その ID がログに記録されるため、同じ ID をたどることで、メール1通1通の配送履歴を追うことができます。

/var/log/maillog の例

```
Jul 26 18:42:23 h006 postfix/master[1195]: daemon started -- version
2.10.1, configuration /etc/postfix
Jul 27 19:19:54 h006 postfix/postfix-script[1160]: starting the
Postfix mail system
Jul 27 19:19:54 h006 postfix/master[1162]: daemon started -- version
2.10.1, configuration /etc/postfix
Jul 28 08:25:36 h006 postfix/postfix-script[1248]: starting the
Postfix mail system
Jul 28 08:25:36 h006 postfix/master[1250]: daemon started -- version
2.10.1, configuration /etc/postfix
Jul 28 15:07:39 h006 postfix/pickup[6595]: D97BDBA: uid=0
from=<root@h006.s142.la.net>
Jul 28 15:07:39 h006 postfix/cleanup[6619]: D97BDBA:
messageid=<20150728060
739.D97BDBA@h006.s142.la.net>
Jul 28 15:07:39 h006 postfix/qmgr[1252]: D97BDBA:
from=<root@h006.s142.la.net>, size=465, nrcpt=1 (queue active)
Jul 28 15:07:39 h006 postfix/local[6621]: D97BDBA:
to=<student@h006.s142.la.net>, orig_to=<student@h006.s142.la.net>,
relay=local, delay=0.09, delays=0.06/ 0.02/0/0.01, dsn=2.0.0,
status=sent (delivered to mailbox)
Jul 28 15:07:39 h006 postfix/qmgr[1252]: D97BDBA: removed
```

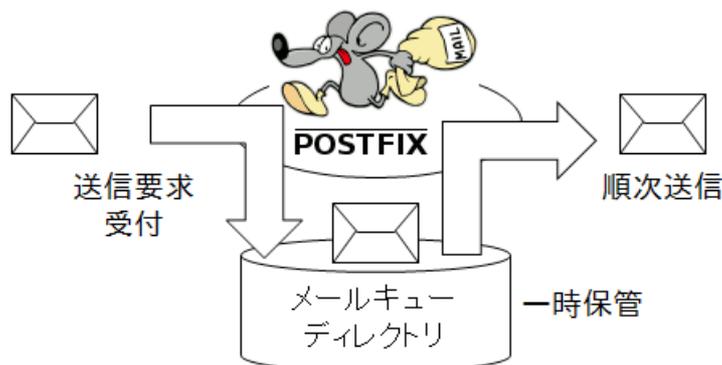
【練習】

ログファイルを確認し、メール配送の履歴を確認します。

4.3 Postfix の運用管理

4.3.1 キューの確認

Postfix はメール配信要求を受け付けると、受け付けたメールを一旦メールキューと呼ばれる場所に格納します。そして、格納されている送信待ち状態のメールを順次送信します。メールは、メールキューディレクトリと呼ばれるディレクトリに保存されます。Postfix におけるメールキューディレクトリは、`/var/spool/postfix/`以下のディレクトリになります。



メールサーバーを運用していると、宛先のメールサーバーに宛先ユーザー名が存在しなかったり、ネットワークのトラブルによって、送信が完了しなかったりということが多々あります。このような場合、再度メールキューディレクトリに登録された後、何度か再送信を試みます。再試行にも失敗した場合は、そのメールを送信したユーザーの元に不達通知が届きます。

メールキューの状況は `mailq` コマンドを利用して、調べることができます。

```
# mailq
-Queue ID- --Size-- ----Arrival Time---- -Sender/Recipient-----
0073EBA*      457 Tue Jul 28 15:23:57  student@h006.s142.la.net

student@example.net
-- 0 Kbytes in 1 Request.
```

Postfix は一定時間ごとに自動的にメールキューに溜まっているメールの処理を行いますが、Postfix コマンドに引数「flush」を指定して実行すると、直ちにメールキュー内のメールの再送信を試みます。

```
# postfix flush
```

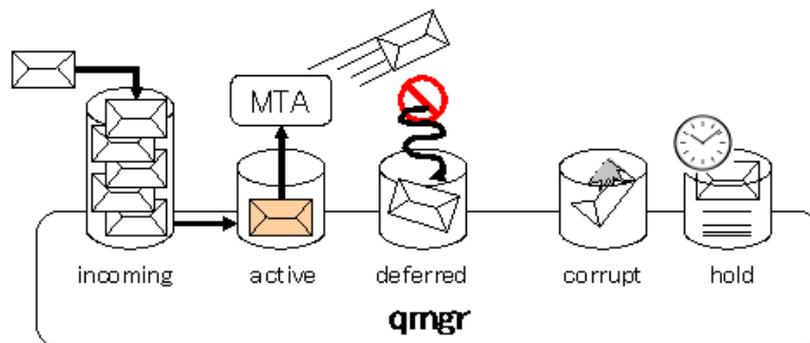
メールキューに保存されているメールをすぐ配信停止したい場合には、`mailq` コマンドで表示された Queue ID の番号を引数に、`postsuper` コマンドを実行して、メールキューからメールを削除します。

```
# postsuper -d 0073EBA
postsuper: 0073EBA: removed
postsuper: Deleted: 1 message
```

【練習】

1. 実際に存在しないユーザーに向けてメールを送ることにより、一時的にわざと不達メールを作り出します。mail コマンドを利用して、student@example.net 宛にメールを送信します。
2. root ユーザーで mailq コマンドを実行し、メールキューにメールが保存されていることを確認します。送信待ちになっているメールの Queue ID を調べます。
3. /var/spool/postfix ディレクトリの中から、実際に送信待ちになっているメールのファイルを探します。
4. 送信待ちになっているメールをメールキューから削除します。
5. mailq コマンドを実行し、メールキューから送信待ちのメールが消えていることを確認します。

/var/spool/postfix/内には、数多くのディレクトリが存在しています。メッセージがはじめに入るメールキューが incoming です。その後 qmgr により、active に移され配送処理されます。何らかの原因で配送処理できなかったメッセージは deferred に格納されます。これ以外に、解析できない・壊れているといったメッセージは corrupt に、特定の条件設定で保留したメッセージは hold へ保持されます。これらのメールキューは qmgr が管理しています。



4.4 転送処理

転送処理は大きく分けて2種類あります。1つは管理者による「システムでの転送処理」、もう一つは「ユーザーによる転送処理」です。

4.4.1 システムでの転送処理

例えば、root 宛のメールをユーザー student が受け取れるようにしたい場合や、システムに存在しないユーザーアカウント lastudent 宛のメールをユーザー student が受け取れるようにしたい場合などには、メールエイリアス(別名)と呼ばれる設定を行います。メールエイリアスの設定は、/etc/aliases ファイルで行います。次にデフォルトの/etc/aliases の内容を示します。

```
# Basic system aliases -- these MUST be present.
mailer-daemon:  postmaster
postmaster:    root

# General redirections for pseudo accounts.
bin:           root
daemon:       root
...
```

メールエイリアスファイルは以下の形式となっています。

```
# コメント
別名宛先:      転送先
```

例えば、次の3行は

```
mailer-daemon:  postmaster
postmaster:    root
root:          student
```

ローカル配送メールのうち、

1. 「mailer-daemon」宛のメールは「postmaster」宛に転送する
2. 「postmaster」宛のメールは「root」宛に転送する
3. 「root」宛のメールは「student」宛に転送する

という意味なり、結果として mailer-daemon 宛のメールは student へ転送されます。

mailer-daemon や postmaster は/etc/passwd ファイルに実在するユーザーではありません。これらのメールアドレスは、主にメールサーバープログラムなどが、エラー通知やログ通知を行う際に指定するものです。最終的に root ユーザーが受け取るようになっているので、実際に配送されたメールを管理者が読むことができるようになっています。

しかし、bin や daemon などのユーザーは実際にシステム上に存在するユーザーです。これらのシステ

ムアカウントはカーネルやプログラムが用いるもので、人間がログインするために用いられるものではありません。これらのユーザーに対して送られたメールを管理者が読めるように、デフォルトのエイリアスファイルで次のような設定がなされています。

```
bin:          root
daemon: root
```

上の設定により、bin や daemon といったユーザー宛のメールは root 宛に配送され、システム管理者が読めるようになります。

よく企業などのホームページに見られる、「info@企業のドメイン名」というメールアドレス宛に送られたメールを、システムに存在するメール処理担当者 alice、bob という2人のユーザーが受け取れるように指定したい場合は、2名のユーザー名を「,」（カンマ）で区切って指定します。

```
info:  alice, bob
```

ただ、Postfix は、この/etc/aliases ファイルを直接参照しません。バイナリ化されたエイリアスデータベースファイル/etc/aliases.db ファイルを参照して、実際の転送先を知ります。テキストファイルである/etc/aliases を編集したら、必ず、/etc/aliases.db を再生成する必要があります。このためには、newaliases コマンドを用います。その際、Postfix を再起動する必要はありません。

```
# newaliases
```

【練習】

1. root ユーザー宛のメールがユーザー student 宛に転送されるように/etc/aliases ファイルの編集を行います。
2. newaliases コマンドで/etc/aliases.db ファイルに反映させます。
3. mail コマンドで root ユーザー宛のメールを送信します。
4. 3.で送信したメールがユーザー student 宛てに届いていることを確認します。

4.4.2 ユーザーによる転送処理

/etc/aliases ファイルによって、転送処理が実行できることがわかりました。しかし、このファイルは/etc/内に存在するファイルです。変更する権限は管理者である root のみが持っていますので、各ユーザーが、システムの自分のアカウント宛に届いたメールを、どこか他のメールサーバーにある全く別のアカウント宛に、転送するように指定することはできません。

そこで、システムでの転送処理とは別に、ユーザーごとに個別の転送設定ができるような機能が用意されています。

設定は以下の通り、ユーザーごとに自らのホームディレクトリ内に.forward ファイルを配置します。このファイルの中に1行1転送先として、宛先のメールアドレスを記述しておきます。これで、このメールサーバーに自分宛のメールを受信するためにアクセスする必要はなくなります。

もちろん、このマシンのアカウント宛のメールも確認できるようにしておくことは可能です。その場合は、このマシン上の自分のメールアドレスを.forward ファイル内に記述しておきます。その際、ループ処理に陥るのを防ぐために、このマシンの自分のメールアドレスの行の先頭に「バックスラッシュ」を記述しておきます(実際には Postfix にループ回避機能があります)。

.forward の例:

```
abi@abc.net  
\student@h006.s142.la.net
```

[練習]

student あてのメールを自身のメール (Web メールを作成) へ転送する設定を行い、動作を確認します。

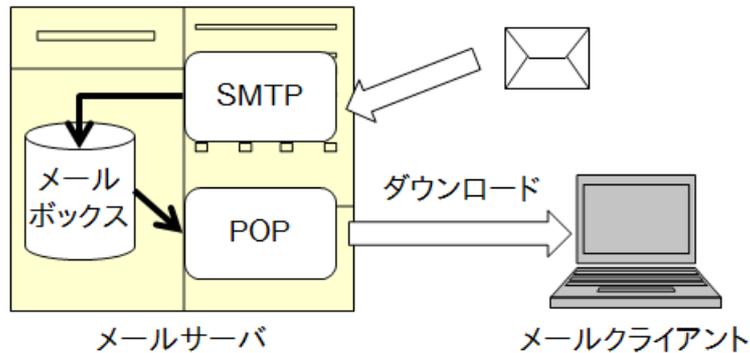
5 メールサーバー (3)

5.1 POP サーバーの役割

POP はメールサーバーに届いた自分宛のメールをメールクライアントにダウンロードするために用いられるプロトコルです。メールサーバー側でユーザーからのメールの受信要求を待ち受けるのが、POP サーバーです。POP は様々な RFC (918, 937, 1081 など) で規定されていますが、今なお改定され続けているプロトコルです。今回はこの POP サーバーを構築して、受信要求を受け付けるメールサーバーを構築します。

5.1.1 メールボックス

SMTP サーバーが受け取ったメールを格納しておく場所をメールボックスと言います。下図のように SMTP サーバーがメールを受け取ると、SMTP サーバーは受け取ったメールをメールボックスに格納します。POP サーバーは、メールクライアントからの要求に応じて、メールボックスからメールを取り出し、メールクライアントに送信します。



ユーザー宛に届いたメールは、ディレクトリ `/var/spool/mail/` 以下のユーザー名と同じ名前のファイル内に保存されます。例えば、ユーザー `student` の場合は、ファイル `/var/spool/mail/student` にメールデータが格納されます。

このファイルを参照すると、次のようになっています。複数のメールが1つのファイル内に格納されていることがわかります。

```
(抜粋)
Content-Transfer-Encoding: 7bit
Message-Id: <20150728060739.D97BDBA@h006.s142.la.net>
From: root@h006.s142.la.net (root)

This is test mail

From root@h006.s142.la.net Tue Jul 28 15:55:03 2017
Return-Path: <root@h006.s142.la.net>

Date: Tue, 28 Jul 2017 15:55:03 +0900
To: student@h006.s142.la.net
Subject: hello
User-Agent: Heirloom mailx 12.5 7/5/10
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Message-Id: <20150728065503.49F0ABA@h006.s142.la.net>
From: root@h006.s142.la.net. (root)

Hello, student
```

POP サーバーは、TCP ポート 110 番でクライアントからのメール受信要求を待ち受けます。メールクライアントからの要求を受け取ると、`/var/spool/mail/ユーザー名`ファイルのメールデータを読み出して、クライアントに送信します。

5.2 Dovecot を使った POP サーバーの起動

5.2.1 Dovecot のインストール

Dovecot は Timo Sirainen が開発公開している、POP3・IMAP 機能を提供する MRA (受信用メールサーバ) で、セキュリティ対策や IPv6 にも対応しており、近年利用される機会が多いソフトウェアです。

CentOS では MRA として Dovecot が標準提供されています。yum を使ってインストールしてください。

```
# yum -y install dovecot
```

【練習】

1. Postfix の動作を確認します。
2. dovecot パッケージがインストールされていることを確認します。インストールされていない場合はインストールしてください。
3. パッケージ dovecot によってインストールされたファイルの一覧を表示します。

5.2.2 Dovecot の設定

設定ファイル/etc/dovecot/dovecot.conf ファイルで設定項目 protocols を編集します。

```
...
# Protocols we want to be serving.
#protocols = imap pop3 lmtp
protocols = pop3                                ←追加
...
```

Dovecot は imap, pop3, lmtp に対応していますが、今回 pop3 以外は利用しません。他のプロトコルは削除します。

続いて、/etc/dovecot/conf.d/10-mail.conf のメールボックスとアクセス権の設定を行います。

```
(省略)
# mail_location = maildir:~Maildir
# mail_location = mbox:~/mail:INBOX=/var/mail/%u
# mail_location = mbox:/var/mail/%d/%1n/%n:INDEX=/var/indexes/%d/
%1n/%n
#
#mail_location =
mail_location = mbox:~/mail:INBOX=/var/mail/%u    ←追加
(省略)
#mail_access_groups =
mail_access_groups = mail                          ←追加
(省略)
```

5.2.3 Dovecot の起動

Dovecot の起動、停止は他のサービス同様 `systemctl` コマンドで行います。

Dovecot の起動

```
# systemctl start dovecot
```

Dovecot の停止

```
# systemctl stop dovecot
```

Dovecot の再起動

```
# systemctl restart dovecot
```

【練習】

1. 設定ファイル(/etc/dovecot/dovecot.conf)を編集し、POP3 プロトコルをサポートするようにします。
2. 設定ファイル(/etc/dovecot/conf.d/10-mail.conf)を編集し、メールボックスに Dovecot がアクセスできるようにします。
3. Dovecot を起動します。
4. telnet で localhost の TCP110 番ポートにアクセスします。
5. POP プロトコルの USER コマンドと PASS コマンドを使って、ユーザー student でログインします。
6. LIST コマンドを使って、届いているメールの一覧を取得します。
7. RETR コマンドを使って、届いているメールのうち1つを取得します。
8. QUIT コマンドで終了します。

5.3 メールクライアントの導入

5.3.1 Linux 上でのメールクライアント

Linux でも Windows と同様に、数々のメールクライアントソフトを利用できます。Windows 上では GUI のメールクライアントソフトが多く用いられています。Microsoft の Outlook や Becky! Internet Mail、Mozilla プロジェクトの Thunderbird などのソフトを普段使っている方も多いことでしょう。

Linux 上では、端末上で動作する CUI メールクライアントと、X Window System 上で動作する GUI メールクライアントがあります。CentOS では Thunderbird が標準提供されていますが、今回は導入手順の解説も兼ねて Sylpheed を使用します。

Sylpheed のホームページ (<http://sylpheed.sraoss.jp>) にアクセスし、ダウンロードのリンクから「ソースパッケージ」をダウンロードし、展開後、./configure を実行します。※1

【練習】

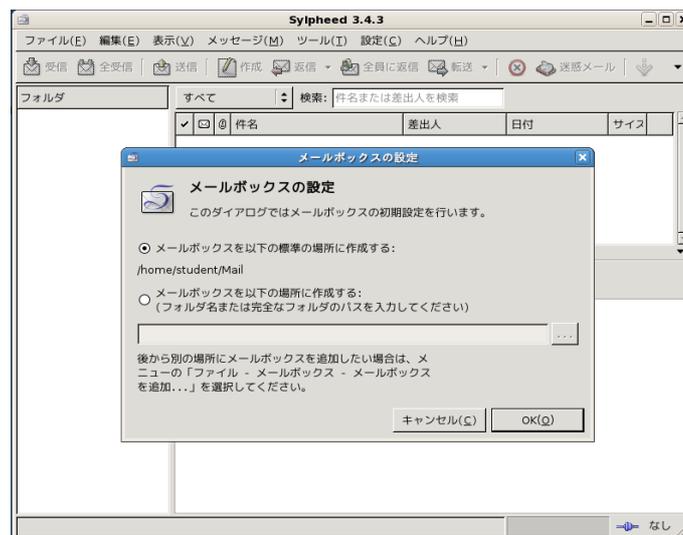
講師の指示に従って Sylpheed のパッケージを入手してインストールを行ってください。

5.3.2 Sylpheed の起動と設定

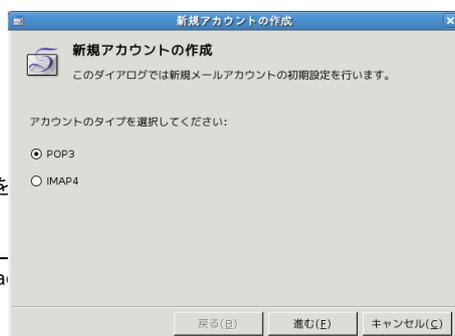
Sylpheed を起動するには、X Window System 上の端末で sylpheed コマンドを実行します。

```
$ sylpheed &
```

またはデスクトップのメニューから「アプリケーション→インターネット→Sylpheed」を選択しても構いません。初めて Sylpheed を起動すると自動的に「メールボックスの設定」画面が表示されます。



Sylpheed で受信したメールの格納先を指定します。規定値はホームディレクトリ下の、Mail ディレクトリの中に保存されるようになります。



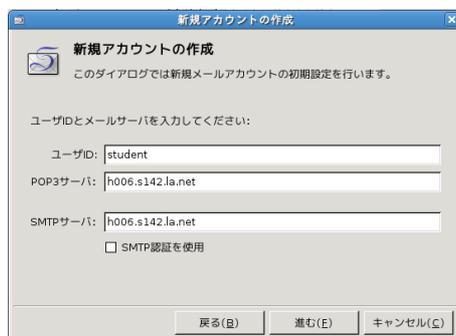
次に受信に使うプロトコルを POP、IMAP の中から選択します。今回は POP を選択します。

※1 configure を

ージをインストールしてください

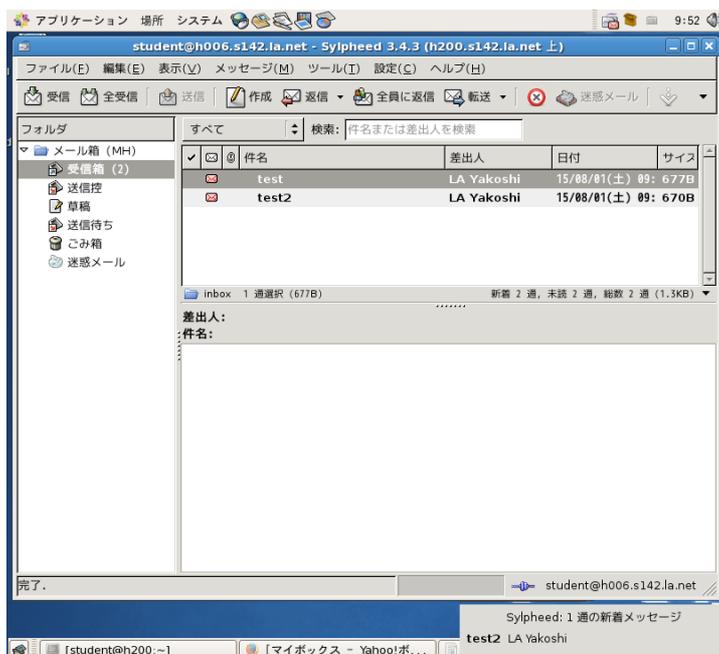


表示名とメールアドレスを登録します。表示名は任意で、メールアドレスは「student@自ホスト名」を登録します。



ユーザー名とサーバー名を入力します。ユーザー名は student、サーバーは POP,SMTP とも自ホストを指定します。

以上で設定は完了です。自分自身へメールし、[受信]ボタンをクリックしメールを取り込みます。



[練習]

1. Sylpheed を起動します。
2. 前述の設定方法を参考に Sylpheed の設定を行います。
3. 隣の方とメールの送受信を行ってみます。

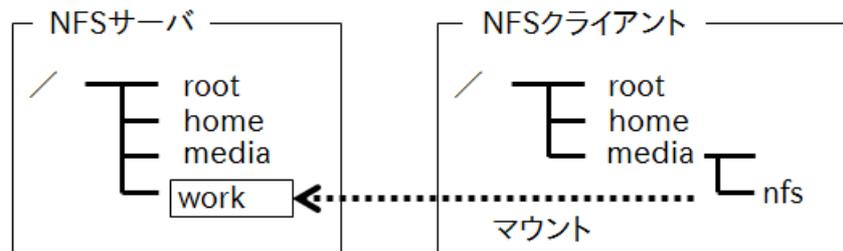
6 NFS

6.1 NFS の概要

NFS (Network File System) は Sun Microsystems 社 (現 Oracle 社) が開発・公開した、UNIX 互換 OS 用のファイル共有サービスです。NFS によるファイル共有は、リモートのファイルシステムをマウントするという形で行われます。

6.1.1 NFS の仕組み

ファイルシステムを公開する側のホストには NFS サーバーが稼働しており、NFS クライアントから NFS サーバーが公開しているディレクトリをマウントすることにより、アクセスすることが可能となります。つまり、NFS サービスはクライアント・サーバー方式によって実現されています。



NFS クライアントは NFS サーバーにマウント要求を送信し、リモートホストとローカルホスト間での接続を確立します。接続を確立した後、NFS サーバーは NFS クライアントからの要求に応じて、ファイルのデータを送信したり、ディレクトリ内のファイル一覧を送信したりします。ネットワーク越しのファイル共有を行うためには、このように必要に応じて、NFS サーバー側から NFS クライアント側に逐一データが送信される必要があります。

6.1.2 NFS サーバープログラム

NFS サービスは以下のプログラムを軸として実現されます。

プログラム	機能
rpc.nfsd	NFS クライアントからの RPC 要求をローカルなファイルシステムでの命令に変換する
rpc.mountd	nfsd からの指示に従い NFS クライアントに対して、ファイルなどのデータを送信する

これらのプログラムは nfs-utils パッケージに含まれ、/usr/sbin/ディレクトリ以下に配置されています。

rpc.nfsd や rpc.mountd は名前の通り RPC プログラムと総称されるプログラムの1つです。RPC (Remote Procedure Call) は、ネットワークを介してリモートホストの様々な機能と呼出すためのインターフェイスです。RPC プログラムを利用するためには portmapper と呼ばれるサービスが必要になります。^{※1}

※1 最新の NFS Ver.4 ではポート番号が固定のため portmapper は不要

CentOS7 では NFS Ver.4 が採用されており、以下のサービスから構成されています。

```
$ systemctl list-unit-files --type=service | grep nfs
nfs-blkmap.service          disabled
nfs-config.service         static
nfs-idmap.service          static
nfs-idmapd.service         static
nfs-lock.service           static
nfs-mountd.service         static
nfs-rquotad.service        disabled
nfs-secure-server.service  static
nfs-secure.service         static
nfs-server.service         disabled
nfs-utils.service          static
nfs.service                disabled
nfs.lock.service           static
```

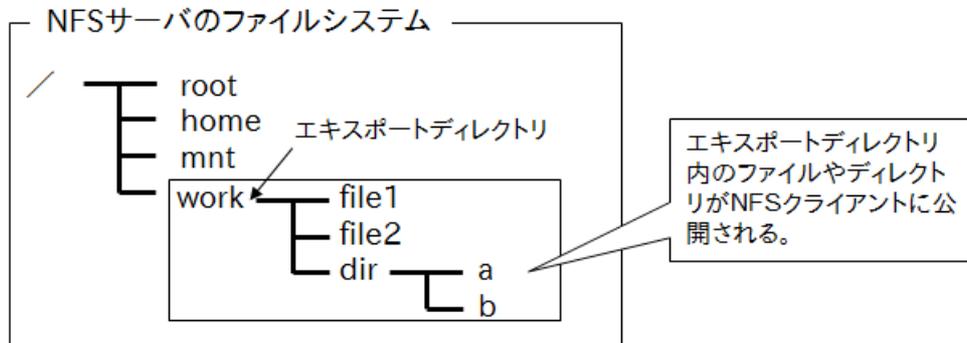
NFS サーバーで、systemctl コマンドを使って nfs-server を起動します。クライアントについては特にサービスを起動する必要はありません。

```
# systemctl start nfs-server
# systemctl status nfs-server
●nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled; vendor
  preset: disabled)
   Active: active (exited) since 火 2017-07-28 21:41:39 JST; 1min 57s ago
   Process: 11809 ExecStart=/usr/sbin/rpc.nfsd $RPCNFSDARGS (code=exited,
  status=0/SUCCESS)
   Process: 11823 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
  Main PID: 11824 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/nfs-server.service

 7月 28 21:41:39 localhost.localdomain systemd[1]: Starting NFS server and s...
 7月 28 21:41:39 localhost.localdomain systemd[1]: Started NFS server and se...
Hint: Some lines were ellipsized, use -l to show in full.
```

6.2 エクスポートの設定と NFS マウント

NFS クライアントにファイルシステムの一部を公開することをエクスポートと呼びます。NFS クライアントに公開するディレクトリの頂点をエクスポートディレクトリと呼びます。エクスポートディレクトリ以下のファイルやディレクトリは、全て NFS クライアントに公開されます。



エクスポートは、設定ファイル/etc/exports の中で設定します。

6.2.1 エクスポートの定義

ファイル/etc/exports には、基本的に1行1つのエクスポートの設定を記述します。例えば、/work というディレクトリを作成し、この/work ディレクトリをエクスポートディレクトリとして 10.20.0.0/255.255.0.0 のネットワークセグメントに公開する場合には、以下のように記述します。

例: /work を 10.20.0.0/255.255.0.0 のネットワークセグメントにエクスポート (公開) する

/work	10.20.0.0/255.255.0.0()	
エクスポートディレクトリ	マウント許可範囲	公開オプション

マウントを許可する範囲は、特定のホストでも構いません。なお公開オプション () との間に空白は入れないでください。

6.2.2 エクスポート設定の反映と確認

前述の設定を有効にするためには、exportfs コマンドに「-a」オプションを指定して実行します。なお、これを実行するためには、事前に NFS サービスが起動している必要があります。

また、エクスポート設定を変更した場合は、exportfs コマンドに「-r」オプションを渡して、変更の反映を行う必要があります。しかし、NFS サービスの再起動は必要ありません。

```
# exportfs -a
```

これにより、/work ディレクトリのエクスポートが実行されました。エクスポートされていることを確認するためには、exportfs コマンドに「-v」オプションを指定して実行します。

```
# exportfs -v
/work          10.20.0.0/255.255.0.0(ro,wdelay,root_squash,no_subtree_check,
sec=sys,ro,secure,root_squash,no_all_squash)
```

確認の表示を見るとエクスポートオプションのカッコ内にいくつかの設定値が入っているのがわかります。

これらは、エクスポートオプションの指定が空の時に自動的に適用されるデフォルトのエクスポートオプションです。

6.2.3 NFS マウント

実際にエクスポートされたディレクトリをマウントしてみましょう。まず、マウントポイントとしてディレクトリ/media/nfsを用意します。

```
# mkdir /media/nfs
```

NFS によりエクスポートされたディレクトリをマウントするためには、mount コマンドを使います。ファイルシステムのタイプは nfs です。

```
mount -t nfs NFS サーバー : エクスポートディレクトリ マウントポイント
```

実際にエクスポートディレクトリをマウントします。マウントできたかどうか df コマンドで確認します。

```
# mount -t nfs h006.s142.la.net:/work /media/nfs
# df /media/nfs
ファイルシステム              1K-ブロック   使用   使用可  使用%  マウント位置
h006.s142.la.net:/work      1020672 65664 955008    7% /media/nfs
```

サーバー側では、showmount コマンドにより、NFS クライアントの利用状況を把握することができます。「-a」オプションにより、クライアントの名前と、ディレクトリが表示されます。^{※2}

```
# showmount -a
All mount points on h006.s142.la.net:
10.20.142.200:/work
```

/work ディレクトリが/media/nfs ディレクトリにマウントされていることが上の表示から確認できます。実際に/work ディレクトリ内にファイルを作成し、/media/nfs に移動してみると、同じファイルを確認することができます。

オリジナルのエクスポートディレクトリでファイルを作成し、確認

```
# touch /work/nfstest
# ls -l /work
合計 0
-rw-r--r-- 1 root root 0  7月 30 08:23 nfstest
```

NFS マウントポイント側も確認

```
# ls -l /media/nfs
合計 0
-rw-r--r-- 1 root root 0  7月 30 08:23 nfstest
```

6.2.4 NFS アンマウント

マウント中のファイルシステムをアンマウントするためには、umount コマンドを使います。DVD-ROM な

※2 showmount コマンドは NFS Ver.3 においては情報が表示されますが、NFS Ver.4 においては表示されません

どを使用するときと同様に、umount コマンドの引数にマウントポイントを指定して実行します。

```
# umount /media/nfs
```

アンマウントの確認も df コマンドで行います。NFS サーバー側も利用者によるマウントが解除されたことを確認します。

```
# showmount -a
All mount points on h006.s142.la.net:
```

[練習]

ディレクトリ/work を作成して、エクスポートを行います。

1. NFS サービスを起動します。必要に応じパッケージを導入します。
2. ディレクトリ/work を作成します。
3. 設定ファイル/etc/exports を編集し、ディレクトリ/work を「10.20.0.0/255.255.0.0」に対してエクスポートできるように設定します。
4. exportfs コマンドを用いて、/etc/exports の設定を有効にします。
5. マウントポイントとなるディレクトリ /media/nfs を作成します。
6. 隣の方のホストで、自分のホストでエクスポートしている/work を/media/nfs にマウントしてもらいます。
7. 隣の方のホストで、df コマンドを用いてマウントが正常に行われたことを確認してもらいます。
8. 隣の方のホストで、/media/nfs ディレクトリをアンマウントしてもらいます。

6.2.5 エクスポートオプション

ファイルシステムをマウントする際には、一般的に書き込み可能 (read-write) などのマウントオプションを指定する必要があります。/etc/exports ファイルには、これらのマウントの際のオプションを指定することができます。エクスポートオプションはマウントオプションよりも優先されます。

主なエクスポートオプション

オプション	解説
ro	クライアントからの読み込みのみ許可する (デフォルト)
rw	クライアントに読み書きを許可する。クライアント側で ro を指定することも可能である
async	書き込みが完了しなくてもクライアントからの次の要求に応じる
sync	書き込みが完了するまでクライアントからの次の要求に応じない (デフォルト)
wdelay	複数の書き込み処理を一度に行う (デフォルト)
no_wdelay	複数の書き込み処理を一度に行わない。ただし、sync が有効なときだけ反映される

上記のオプションの中で特に重要なものは ro と rw です。エクスポートディレクトリに対する書き込み権限が可能かどうかを指定するもので、デフォルトで ro (read-only) が指定され、書き込みは不可能になっています。

NFS サーバー側のファイルへの書き込みを許可するエクスポートを定義するためには、以下のよう
に/etc/exports ファイルで指定します。

```
/work      10.20.0.0/255.255.0.0(rw)
```

既存のエクスポートの設定を変更し、再度エクスポートを行うためには、exportfs コマンドに「-r」オプションを指定して実行します。

```
# exportfs -r
# exportfs -v
/work      10.20.0.0/255.255.0.0(rw,wdelay,root_squash,no_subtree_check, sec=sys,
rw,secure,root_squash,no_all_squash)
```

このようにした後に、実際にエクスポートディレクトリをマウントすると、実際にファイルの書き込みが可能になります。

【練習】

/tmp ディレクトリは誰でも書き込み可能に設定されています。以下では、/tmp ディレクトリを書き込み可能でエクスポートし、これをマウントします。

1. /etc/exports ファイルに、/tmp ディレクトリを教室のネットワークセグメントから書き込み可能でマウントすることを許可する設定をします。
2. 「exportfs -r」を実行し、エクスポートの設定を反映させます。
3. ローカルホスト上でエクスポートされている/tmp ディレクトリを、/media/nfs ディレクトリにマウントします。このとき、マウント実行前に他のファイルシステムが、/media/nfs ディレクトリにマウントされていないことを確認します。
4. /media/nfs ディレクトリに移動し、「touch write.test」を実行して、空のファイルを作成してみます。エラーがなく、実行できたことを確認したら、/tmp ディレクトリに移動し、ファイルの作成を確認します。
5. /media/nfs ディレクトリをアンマウントします。

6.3 NFS マウントにおけるユーザー権限

他のホストのファイルシステムへのアクセスを提供する NFS サービスでは、ユーザー権限はどのように管理されているのでしょうか？たとえば、クライアント側の student というユーザーは NFS サーバー側ではどのようなユーザーとして扱われるのでしょうか？

書き込みを許可する場合には特にこのユーザー権限の扱いが重要になります。

6.3.1 UID、GID によるマッピング

NFS では2つのホスト間のユーザーは UID と GID を使って区別されます。NFS クライアント上のユーザー student の UID が 1000 であった場合、NFS サーバー側では UID が 1000 の自ホストのユーザーであると認識されます。つまりユーザー名が違っていても、UID が一致すれば、同じユーザーであるとみなします。グループに関してもこれと同様で、GID によって、個々のグループを認識します。

UID マッピングの例

NFS クライアント上のユーザーアカウント		NFS サーバー上のユーザーアカウント
UID 1000 - student	←→	UID 1000 - student
UID 1001 - teacher	←→	UID 1001 - staff
UID 1600 - staff	←→	UID 1600 - linus
UID 1777 - penguin	←→	不在

上の例のように、UID とユーザー名の対応関係が NFS クライアント側と NFS サーバー側とで一致しているとは限りません。ユーザー student に関しては UID が一致して同じユーザーと認識されても、teacher と staff が同じユーザーであると認識されるといった厄介な現象が起きる可能性があります。これを防ぐには、/etc/passwd ファイルや/etc/group ファイルなどのアカウント情報が記述されたファイルを双方のホスト間で同一のものを用いるという方法があります。

また、規模の大きなネットワーク内であれば、ユーザーアカウントを一元管理するパスワードサーバーなどを配すことにより、常に同一のユーザーとしての割り当てが行われ、問題を解消することもできます。

6.3.2 root ユーザーのマッピング

root ユーザーの UID は、どの UNIX 系 OS でも同じで、0 です。つまり root ユーザーに関しては、どの UNIX 系 OS でも必ず root としてマッピングされるということになります。

しかし、root ユーザーは大変強い権限を持っているので、他のホストの root ユーザーを自ホストの root ユーザーにマッピングするのは非常に危険です。

root_squash オプションはデフォルトで有効で、root ユーザーからのアクセスは全て別のユーザー（匿名ユーザー）に置き換えます。CentOS では nfsnobody (UID=65534) というユーザーにマッピングされます。

このことを確かめるために、前節で作成した/tmp ディレクトリをマウントし、このディレクトリの中に NFS クライアント側の root ユーザーでアクセスして、空のファイルを作成してみます。

```
# mount -t nfs 10.20.142.6:/tmp /media/nfs
# touch /media/nfs/rootwrite
# ls -l /media/nfs/rootwrite
-rw-r--r-- 1 nfsnobody nfsnobody 0 7月 30 09:05 /media/nfs/rootwrite
```

上の実行結果から「nfsnobody」というユーザー名とグループ名が表示されているファイルが、NFS クライアント側で root によって作成されたファイルであることがわかります。NFS クライアント側の root ユーザーを NFS サーバー側でも root としてマッピングすることを許可するためのオプションが「no_root_squash」です。

```
/tmp          10.20.0.0/255.255.0.0(rw,no_root_squash)
```

これにより root_squash オプションが無効化されます。実際にこの設定を反映させてマウントすると、今度は root ユーザーとして処理が行われていることが確認できます。

```
# umount /media/nfs
# exportfs -r
# mount -t nfs 10.20.142.6:/tmp /media/nfs
# touch /media/nfs/rootwrite2
# ls -l /media/nfs/root*
-rw-r--r-- 1 nfsnobody nfsnobody 0 7月 30 09:05 /media/nfs/rootwrite
-rw-r--r-- 1 root      root      0 7月 30 09:08 /media/nfs/rootwrite2
```

6.3.3 その他のエクスポートオプション

ユーザー権限のマッピングに関するエクスポートオプションとしては、root ユーザー以外の一般ユーザーも nfsnobody にマッピングさせるオプションや、UID や GID をどのユーザーアカウントとマッピングさせるか設定できる(デフォルトで nfsnobody)ものなども用意されています。

オプション	解説
all_squash	一般ユーザーアカウントの全てを匿名ユーザーにマッピングさせる
anonuid	匿名ユーザーの UID 指定(デフォルトは 65534=nfsnobody)
anongid	匿名ユーザーの GID 指定(デフォルトは 65534=nfsnobody)
no_subtree_check	パフォーマンス向上のためファイル位置のチェック機能を使わない

6.4 確認問題

1. 現在実行されている全ての NFS マウントをアンマウントします。
2. /home ディレクトリを教室のネットワークセグメント内のホストに書き込み可能でエクスポートします。また、root ユーザー が nfsnobody にマッピングされるように root_squash オプションを有効にします。
/etc/exports ファイルに必要な設定を記述します。
3. exportfs コマンドで設定変更を反映させます。
4. 準備が完了したら、隣の方に/home ディレクトリを、/media/nfs ディレクトリにマウントしてもらいます。
5. お互いのホストの/etc/passwd ファイルを参照し、UID とユーザー名の組み合わせが一致するユーザーを調べます。インストール時に作成した student ユーザーの UID は 1000 で一致しているはずですが、ユーザー名と UID の組み合わせが一致するユーザーが存在しない場合には、useradd コマンドに「-u ユーザー ID」を指定して、新たなユーザーアカウントを作成しましょう。ここで UID を指定することにより、2つのホストで UID とユーザー名が一致するユーザーを作成することができます。
6. UID とユーザー名が一致するユーザーで、/media/nfs ディレクトリに移動し、そのユーザーのホームディレクトリ内を参照します。
7. ホームディレクトリ内に空のファイルを touch コマンドで作成します。
8. ls コマンドを使って、作成したファイルの所有者を確認します。
9. 別のユーザーのホームディレクトリ内への移動を試みます。パーミッションの関係で処理が実行できないことを確認します。
10. /media/nfs ディレクトリをアンマウントします。

7 FTP

7.1 オープンソースソフトウェアの利用

Linux はオープンソースソフトウェア (OSS) で構成されています。OSS はソースコードが公開されています。そのため利用者が用途によってソフトウェアに改訂を加え、自分専用にカスタマイズすることが可能です。本章では、CentOS 標準の FTP サーバーソフト (vsftpd) とは異なる FTP サーバーソフト (ProFTPD) を、ソースコードから利用する方法を確認していきます。

7.1.1 パッケージの入手

OSS は一般的に tarball (圧縮した tar 形式のファイル) として、インターネット上の FTP サイト (本家・ミラーサイトを含む) に公開されています。使用したいソフトウェアの tarball をこれらのサイトからダウンロードします。

GUI 環境下なら、ブラウザ上からクリックでダウンロードし、CUI 環境下であれば、`wget` や `ftp` コマンドを利用します。`ftp` コマンドがインストールされていない場合は、`yum` コマンドを使って予めインストールしておいてください。

ProFTPD は、本家サイトが www.proftpd.org、国内ミラーサイトは複数ありますが、今回は ftp.riken.jp の例を示します。^{※1}

例: `ftp.riken.jp` から ProFTPD ソースパッケージ入手

```
# ftp ftp.riken.jp
Connected to ftp.riken.go.jp (134.160.38.1).
220 ::ffff:134.160.38.1 FTP server ready
Name (ftp.riken.jp:root): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230 Anonymous access granted, restrictions apply
Remote system type is UNIX
Using binary mode to transfer files.
ftp>
```

CUI 環境から `ftp` コマンドで FTP サイトを利用する場合、ユーザー名に「anonymous」または「ftp」。パスワード代わりに「メールアドレス」を入力することで匿名ユーザーとしてログインして、サイトを利用することが可能になります。

※1 `ftp.riken.jp` は同一 IP アドレスからの同時接続数を 3 に制限しています

ログインが完了したら、ディレクトリ構成を確認します。

```
ftp> ls
227 Entering Passive Mode (134,160,38,1,156,229).
150 Opening ASCII mode data connection for file list
drwxr-xr-x  4 root          root          4096 Aug  8  2013 cernlib
          [中略]
drwxr-xr-x 16 root          root          4096 Sep 11  2015 net
drwxr-xr-x 34 root          root          4096 Feb 23  2015 NetBSD
drwxr-xr-x  4 root      root          33 Sep  9  2015 office
          [中略]
226 Transfer complete
```

今回の ProFTPD はこのサイトの場合、「net」というディレクトリ内に ProFTPD としてディレクトリが用意されています。

```
ftp> cd net
250 CWD command successful
ftp> ls
227 Entering Passive Mode (134,160,38,1,177,225).
150 Opening ASCII mode data connection for file list
drwxrwxrwx 210 archive  archive      8192 Aug 14 23:30 apache
          [中略]
drwxrwxr-x  5 archive  archive      85 Aug 14 17:45 ProFTPD
drwxr-xr-x 36 archive  archive     4096 Aug 14 17:40 samba
          [中略]
226 Transfer complete
ftp> cd ProFTPD
250 CWD command successful
ftp> ls
227 Entering Passive Mode (134,160,38,1,138,118).
150 Opening ASCII mode data connection for file list
drwxrwxr-x  3 archive  archive      19 Jul  1  2005 devel
drwxrwxr-x  4 archive  archive      19 Dec  2  2010 distrib
drwxrwxr-x  4 archive  archive      34 Jul  1  2005 historic
-rw-rw-r--  1 archive  archive     451 Jul  1  2005 README.MIRRORS
226 Transfer complete
```

ProFTPD ディレクトリ内にある distrib ディレクトリに、cd コマンドで移動して、ディレクトリ構成を確認します。今回はソースコードを使用しますから、更に source ディレクトリに移動します。

```
ftp> cd distrib
250 CWD command successful
ftp> ls
227 Entering Passive Mode (134,160,38,1,139,185).
150 Opening ASCII mode data connection for file list
drwxrwxr-x  2 archive  archive     4096 Apr 10 03:00 source
226 Transfer complete
```

```
ftp> cd source
250 CWD command successful
```

最新版の tarball をダウンロードします。膨大なファイルが存在している場合もあります。間違えないように注意します。

```
ftp> ls
227 Entering Passive Mode (134,160,38,1,135,60).
150 Opening ASCII mode data connection for file list
-rw-r--r-- 1 archive archive 29988477 May 28 2015 proftpd-1.3.5a.tar.gz
      [中略]
-rw-r--r-- 1 archive archive 20251898 Apr 10 03:00 proftpd-1.3.6.tar.gz
-rw-r--r-- 1 archive archive 204 Apr 10 03:00 proftpd-1.3.6.tar.gz.asc
-rw-r--r-- 1 archive archive 57 Apr 10 03:00 proftpd-1.3.6.tar.gz.md5
226 Transfer complete
```

get コマンドでファイル名を指定してダウンロードします。また、.md5 ファイルもダウンロードします。ダウンロード完了したら、quit コマンドで終了します。

```
ftp> get proftpd-1.3.6.tar.gz
local: proftpd-1.3.6.tar.gz remote: proftpd-1.3.6.tar.gz
227 Entering Passive Mode (134,160,38,1,182,207).
150 Opening BINARY mode data connection for proftpd-1.3.6.tar.gz (20251898 bytes)
226 Transfer complete
20251898 bytes received in 1.78 secs (11350.12 Kbytes/sec)
ftp> get proftpd-1.3.6.tar.gz.md5
local: proftpd-1.3.6.tar.gz.md5 remote: proftpd-1.3.6.tar.gz.md5
227 Entering Passive Mode (134,160,38,1,153,139).
150 Opening BINARY mode data connection for proftpd-1.3.6.tar.gz.md5 (54 bytes)
226 Transfer complete
54 bytes received in 0.00623 secs (8.66 Kbytes/sec)
ftp> quit
221 Goodbye.
```

カレントディレクトリにあるダウンロードしたファイルの大きさがサーバー上と同じであることを確認します。

```
# ls -l proftpd-1.3.6.tar.gz
-rw-r--r-- 1 root root 20251898 8月 15 09:38 proftpd-1.3.6.tar.gz
```

また、正しくダウンロードされたことを確認するには、.md5 ファイルの内容と md5sum コマンドの実行結果を比較します。一致すれば正しくダウンロードされたことになります。

```
# cat proftpd-1.3.6.tar.gz.md5
13270911c42aac842435f18205546a1b proftpd-1.3.6.tar.gz
# md5sum proftpd-1.3.6.tar.gz
13270911c42aac842435f18205546a1b proftpd-1.3.6.tar.gz
```

【練習】

講師の指示に従って最新版の ProFTPD パッケージを入手します。

7.1.2 tarball の展開とドキュメントの確認

ダウンロードしたファイルは、一旦作業用のディレクトリで解凍展開します。通常こういった作業用に /usr/local/src が用意されていますので、そこで解凍展開します。

```
# cd /usr/local/src
# tar xzvf ~/proftpd-1.3.6rc2.tar.gz
      [中略]
```

展開されたディレクトリに移動して、ファイル構成を確認します。

```
# cd proftpd-1.3.6/
# ls
COPYING          README.Solaris2.5x  config.h.in      locale
CREDITS          README.Unixware     config.sub       ltmain.sh
ChangeLog        README.capabilities configure         m4
INSTALL          README.classes      configure.in     modules
      [省略]
```

通常、展開されたディレクトリ内にはインストール作業を解説したファイル「INSTALL」や、付加情報を記述した「README」などが構成されています。SPEC ファイル (proftpd.spec) がある場合、このファイルから RPM パッケージを作成することもできます。

【練習】

ダウンロードした ProFTPD の tarball を解凍展開して、ファイル構成の確認をします。

7.1.3 コンパイル準備

次にコンパイル作業の手順書である「Makefile」を探します。

もし、このファイルがない場合は手順書を生成するためのファイル「configure」を探します。configure ファイルは Makefile の様々な仕様を決めることができるスクリプトファイルです。これらの仕様をオプションとして configure をコマンドに渡して実行することで、仕様を満たす内容の Makefile が生成されます。特に使用頻度の高いオプションが「--prefix=インストール先ディレクトリ」です。

今回は /usr/local/proftpd にインストールするように指定して configure を実行します。
なお gcc がインストールされていないようであれば、yum コマンドを使ってインストールしてください。

```
# ./configure --prefix=/usr/local/proftpd
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking target system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
      [省略]
config.status: creating Makefile
config.status: creating Make.rules
config.status: creating config.h
config.status: executing libtool commands
config.status: executing default commands

-----
Build Summary
-----

Building the following static modules:
    mod_ident
    mod_cap

-----
```

作業中に何の問題もなければ、様々な処理が走った後でプロンプトが戻ってきます。この段階で既に、Makefile やその他の新規ファイルが生成されているはずです。

[練習]

/usr/local/proftpd ディレクトリにインストールするように設定して、Makefile を生成します。

7.1.4 コンパイルとインストール

次に、生成された Makefile を元にコンパイルを行います。コンパイル作業を実行するコマンドが「make」です。make コマンドは Makefile が存在する作業ディレクトリ内で実行します。

```
# make
cd lib/ && make lib
make[1]: ディレクトリ `/home/student/proftpd-1.3.6/lib' に入ります
gcc -DHAVE_CONFIG_H -DLINUX -I.. -I../include -g2 -O2 -Wall -fno-omit-frame-pointer
-c pr_fnmatch.c
In file included from pr_fnmatch.c:260:0:
pr_fnmatch_loop.c: 関数 'internal_fnmatch' 内:
pr_fnmatch_loop.c:75:7: 警告: 変数 'is_seqval' が設定されましたが使用されていません [-Wunused-but-set-variable]
    int is_seqval;
        ^
gcc -DHAVE_CONFIG_H -DLINUX -I.. -I../include -g2 -O2 -Wall -fno-omit-frame-pointer
-c sstrncpy.c
gcc -DHAVE_CONFIG_H -DLINUX -I.. -I../include -g2 -O2 -Wall -fno-omit-frame-pointer
-c strsep.c
[省略]
```

今度も様々な処理が走ります。何も問題がなければ作業終了でプロンプトが戻ってきますが、途中で問題が起こった場合はその処理が中断されて、エラーメッセージが出力されます。問題を解決できれば、再度コンパイルを行えます。

ディレクトリ内の構成をもう一度確認すると、出来上がったばかりのバイナリファイルがいくつも存在しているのがわかります。しかし、これらのバイナリファイルをプログラムとして実行するには、正しいディレクトリ構成の中に再配置しなくてはなりません。この作業は Make コマンドに install という引数を指定して実行します。

```
# make install
cd lib/ && make lib
make[1]: ディレクトリ `/home/student/proftpd-1.3.6/lib' に入ります
make[1]: `lib' に対して行うべきことはありません。
make[1]: ディレクトリ `/home/student/proftpd-1.3.6/lib' から出ます
cd src/ && make src
make[1]: ディレクトリ `/home/student/proftpd-1.3.6/src' に入ります
make[1]: `src' に対して行うべきことはありません。
[省略]
```

【練習】

上記の手順でコンパイルとインストールを行い、インストール先に指定したディレクトリ構成が作成されていることを確認します。

7.2 FTP サーバー

FTP サービスは異なるコンピューター間でファイルのやりとりを行うための仕組みです。ネットワークセグメント内で利用することもあります。主にインターネットを介して様々なファイル共有をサポートするために利用されます。

FTP サーバーには、Pure-FTPd、ProFTPD、vsftpd などいくつかのサーバーソフトがありますが、ProFTPD は Apache の設定ファイルと似た設定が可能であり、細かいアクセス制限が行え、セキュリティへの配慮も行われている FTP サーバーソフトです。

7.2.1 ProFTPD の基本的な設定

ProFTPD の設定は `/usr/local/proftpd/etc/proftpd.conf` で行います。proftpd.conf はディレクティブ形式を採用しているため、Apache と同様、次のどちらかの構造で設定することになっています。

```
# コメント
(1) ディレクティブ名 値
(2) <ディレクティブ名>
    :
    </ディレクティブ名>
```

(1) は、ディレクティブ名に対応する設定値が指定されたものに設定され、(2) はディレクティブ名で囲まれている中で設定が行えます。また、行頭に「#」を記述するとコメント扱いとなり、その行は設定としては無効になります。

proftpd.conf の主な設定 (1) の書式

ディレクティブ	解説(デフォルト)
ServerName	アクセスされたときに表示される文字列を指定する。提供するサービス名に合わせて適切な名前をつける (ProFTPD Default Installation)。
ServerType	standalone か inetd から選択する (standalone)。
RootLogin	root ユーザーのログインを許可するかどうかを指定する。許可する場合は on、許可しない場合は off とする (off)。
User	ProFTPD のプロセスの所有ユーザーを指定する (nobody)。
Group	ProFTPD のプロセスの所有グループを指定する (nogroup)。
Umask	新しく作成されるファイルやディレクトリのパーミッションを指定するためのマスク値を指定する (022)。

proftpd.conf の主な設定 (2) の書式

ディレクティブ	解説
<Directory>	設定対象のディレクトリを指定できる。
<Limit>	FTP コマンドの制限を指定できる。
<Global>	メインサーバーと仮想サーバーに適用される設定を指定できる。
<VirtualHost>	仮想 FTP サーバーの設定を指定できる。複数のホスト名や IP アドレスで運用する際に利用する。
<Anonymous>	匿名 FTP の設定を行う。不特定のユーザーのログインを許可し、ファイルのダウンロードなどを受け付ける。

[練習]

ProFTPD の設定ファイル `/usr/local/proftpd/etc/proftpd.conf` を開きます。ProFTPD のプロセスの所有ユーザーと所有グループの設定値を確認します。所有グループとして設定されている、

「nogroup」を「nobody」に変更してください。

7.2.2 ProFTPD の起動・停止

通常の RPM パッケージによるインストールであれば、起動スクリプトが用意されています。しかし、今回はソースの tarball からコンパイル・インストールしたので、起動は ProFTPD の本体プログラムを直接実行し、停止はこのプログラムのプロセスを kill コマンドで停止させることになります。

本体プログラムは /usr/local/proftpd ディレクトリ内の sbin ディレクトリ内に「proftpd」ファイルとして存在しています。パスが通っていませんから、絶対パスで指定するか、「./proftpd」と相対パスで指定するかのどちらかで実行します。

```
# /usr/local/proftpd/sbin/proftpd
  または
# cd /usr/local/proftpd/sbin/
# ./proftpd
```

起動の確認は「netstat -antp」コマンドでポートと PID の確認を行います。

```
# netstat -antp | grep proftpd
tcp        0      0 0.0.0.0:21          0.0.0.0:*          LISTEN
23037/proftpd: (acc
```

上の例ではポートは21/TCP、PID は 23037 を示しています。

【練習】

ポートの開閉状態や、プロセスの確認を行って、起動している ProFTPD を特定し、kill コマンドで停止させます。

7.2.3 ホストアクセス制限

FTP サーバーを用いると、クライアントからサーバーのファイルをダウンロードや、アップロードすることができますが、無制限に操作ができてしまうのは、セキュリティ上好ましいとはいえません。

したがって、FTP による操作を制限することが必要になります。こんな時、ProFTPD では「Limit」というディレクティブで様々なアクセス制限を設定できます。

```
<Limit STOR>
  DenyAll
</Limit>
```

上の例ではクライアントからサーバーへのファイルのアップロードを全て禁止するというものです。このように、FTP を通じて行う操作を制限することができます。設定を行うには proftpd.conf 内で<Limit コマンド>~</Limit>を用います。

ここで指定できるコマンドには次のようなものがあります。

FTP コマンド

コマンド	解説
RETR	サーバーからクライアントへのファイル転送(ダウンロード)
STOR	クライアントからサーバーへのファイル転送(アップロード)
REN	ファイル名の変更
DELE	ファイルの削除
RMD	ディレクトリの削除
SITE_CHMOD	パーミッションの変更

FTP コマンドグループ

コマンド	解説
DIRS	ディレクトリ内の一覧を取得をする全ての FTP コマンド
READ	ファイルの読み込みをする全ての FTP コマンド
WRITE	ファイルやディレクトリの作成や削除をする全ての FTP コマンド
ALL	全ての FTP コマンド

<Limit コマンド>~</Limit>中では、Allow、AllowAll、AllowUser、AllowGroup、Deny、DenyAll、DenyUser、DenyGroup などを用いてアクセス制限を行うことになります。Allow と Deny の優先順位は、Apache と同様に「Order」ディレクティブで設定できます。^{※2}

例えば、10.20.0.0/255.255.0.0 からのアクセスを除いて、ファイルの書き込み拒否をするには、以下のようになります。

```
<Limit WRITE>
  Allow from 10.20.
  DenyAll
</Limit>
```

【練習】

※2 Order ディレクティブが省略された場合、Deny が Allow より優先されます

1. proftpd.conf を確認して、クライアントからサーバーへのファイル転送(アップロード)が制限されている設定を確認し、ProFTPD を起動させてください。
2. ローカルホストの ProFTPD に接続し、ファイルのアップロードを試みます。
3. proftpd.conf を編集し、10.20.0.0/16(教室のネットワークセグメント)からのアクセスについて、クライアントからサーバーへのファイル転送を許可します。
4. ProFTPD を停止したのち、再度起動します。
5. 再びローカルホストの ProFTPD に接続し、ファイルのアップロードを試みます。
6. cd コマンドでディレクトリの移動を行い、移動できるディレクトリの範囲を確認します。

7.2.4 ディレクトリ制限(chroot)

FTP でログイン後、カレントディレクトリはログインユーザーのホームディレクトリとなっています。しかし、cd コマンドで移動をすることで、許可されている限りどのディレクトリの中であっても、自由に移動することができます。

通常の用途では、この設定で問題ありませんが、特定の用途で FTP を利用したい場合などでは、設定を変更できると便利です。具体的には、特定のディレクトリをルートディレクトリとして設定し、そのディレクトリ内しか利用できないようにします。

例えば、Web ページ更新のために FTP を利用させたい場合、~/public_html をルートディレクトリとしておきます。

こうすることで、ユーザーが不必要なディレクトリを見ることがなくなるため、セキュリティ的に良いといえます。多くのプロバイダはこの仕組みを利用して、Web ページのコンテンツをアップロードするための FTP サーバーを提供しています。

【練習】

1. proftpd.conf の設定を以下のように変更します。

```
#DefaultRoot  ~
↓
DefaultRoot  ~/public_html
```
2. ProFTPD を停止・起動します。
3. student のホームディレクトリに public_html ディレクトリを作成します。
4. 起動した FTP サーバーに ftp コマンドでアクセスし、ログインします。
5. pwd コマンドや ls コマンドでカレントディレクトリの位置および配置されているファイルを確認します。

7.2.5 ユーザー制限

特定のユーザーに対して、FTP でのログインを禁止したい場合があります。例えば、root ユーザーでの FTP サーバーへのログインは禁止することが望ましいと考えられます。これは、FTP が TELNET と同様、ユーザー名とパスワードを平文でネットワークに流してしまうためです。

特定のユーザーに対して、FTP でログインできないように設定するには、`/etc/ftpusers` ファイルに FTP でのログインを禁止するユーザー名を記述します。そうすることで、該当ユーザーはこの FTP サーバーにログインできなくなります。

例:`/etc/ftpusers`

```
root
bin
daemon
mail
```

[練習]

student ユーザーの名を `/etc/ftpusers` に記述し、student としてログインを試みます。ログインに失敗することを確認します。

8 システム管理 (1)

8.1 システムログ

サーバーを構築する上で、サーバーを適切に設定することはもちろん重要ですが、サーバー稼働開始後の運用も非常に大切です。特に Linux カーネルや各サービスが出力する情報は、システムが正しく稼働しているかどうか、外部からの攻撃を受けていないかなどをチェックする上で極めて重要です。

コンピューター上で動作する多くのサーバーソフトウェアは、稼働中に行った処理やエラー情報などを記録しています。この記録のことをログと呼びます。システム管理者は、ログを見ることで、各ソフトウェアがどのような処理を行っているのか、トラブルは発生していないかなどを知ることができます。

既にいくつかのソフトウェアでログについては学んできましたが、ここでは、各ソフトウェアのレベルよりも、もう一歩高い視点で学ぶことにしましょう。

8.1.1 システムログとは

各ソフトウェアによって出力されるログは、多くの場合、システムログガーと呼ばれる仕組みを利用して、ファイルなどに記録されます。システムログガーは、rsyslog というサービスで、ローカルホストのログデータだけではなく、外部ホストからのログメッセージの受信や記録などの機能も持っています。

rsyslog は「reliable (信頼できる) syslog」の略で、従来から使われてきた syslog を改良し大幅にセキュリティ面を向上させています。起動や停止は他のサービス同様 systemctl コマンドを用います。

```
# systemctl stop rsyslog
# systemctl start rsyslog
# systemctl status rsyslog
●rsyslog.service - System Logging Service
   Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor preset:
   enabled)
   Active: active (running) since 木 2017-07-30 11:09:39 JST; 4s ago
   Main PID: 812 (rsyslogd)
   CGroup: /system.slice/rsyslog.service
           └─812 /usr/sbin/rsyslogd -n

 7月 30 11:09:39 h006.s142.la.net systemd[1]: Starting System Logging Servi...
 7月 30 11:09:39 h006.s142.la.net systemd[1]: Started System Logging Service.
Hint: Some lines were ellipsized, use -l to show in full.
```

システム管理の基本となるサービスですので、停止させてしまうことはほとんどありません。

また設定ファイルは/etc/rsyslog.conf ですが、旧来の syslog.conf と互換性が保たれています。

```
# コメント
セレクト (情報の選択)          アクション (出力先)
```

8.1.2 ログのセレクト

rsyslog で取り扱うログデータ (ログメッセージ) は、ファシリティとプライオリティによって分類されます。これをセレクトと呼び、ドット (.) で接続して表記します。ファシリティやプライオリティを複数指定する場合は

カンマ(,)で区切り、セレクト自体を複数記述する場合はセミコロン(;)で区切ります。

例)

uucp,news,crit ファシリティが uucp または news で、プライオリティが crit 以上
*.info;mail.none ファシリティに関わらずプライオリティが info 以上、ただしファシリティが mail は除く

ファシリティとは、そのログがどのカテゴリに分類されるかを表します。ファシリティには、以下のようなものがあります。

主なファシリティ

ファシリティ	解説
authpriv	login や su などの認証システムに関するメッセージ
cron	スケジュール実行 (cron) に関するメッセージ
daemon	各種サービス (daemon) に関するメッセージ
kern	カーネルに関するメッセージ
lpr	プリンタ (line printer) に関するメッセージ
mail	メールに関するメッセージ

プライオリティは、ログの重要度や緊急性のレベルを表します。いわゆる「しきい値」に相当するものです。最も重要度が高いとされる「emerg」から、デバッグの情報を提示してくれる「debug」まで、段階的にプライオリティが準備されています。しきい値として指定されたプライオリティは、それ以上のプライオリティを持つものも必ず出力の対象になります。つまり、ある特定のプライオリティを設定すれば、それ以上のプライオリティも対象になります。

主なプライオリティ(重要度順)

プライオリティ	解説
emerg (panic)	極めて危険な状態 (OS がダウンするレベル)
alert	危険な問題 (早急に対処が必要)
crit	致命的な問題 (ハードウェアトラブル)
err(error)	一般的な問題
warning (warn)	警告
notice	通知
info	情報
debug	デバッグ用情報
none	出力しない (無視する)

8.1.3 ログのアクション

rsyslog は、各ソフトウェアから出力されるログをファシリティとプライオリティごとに、どんな処理を行うかを定めることができます。rsyslog が行う処理のことをアクションと呼びます。アクションにはファイルへの記録や画面への出力などが準備されています。

主なアクション

アクション	解説	例
/ファイルパス	指定されたファイルへ追記	/var/log/messages
/dev/デバイスファイル名	指定されたデバイスへ出力	/dev/tty1

コマンド	コマンドへパイプラインで接続	/bin/cat -n
:omusrmsg:ユーザー名	指定されたユーザーのコンソールに通知	:omusrmsg:student
@ホスト	指定されたサーバーの rsyslog へ転送	@h006.s142.la.net

実際に rsyslog.conf に記載している内容を例として解説します。

```
mail.* -/var/log/maillog
```

syslog.conf では「*」はシェル上のワイルドカードと同様に「すべて」という意味を持ちます。ここでの例では、ファシリティが mail であれば、プライオリティに関わらず（つまりメールシステムに関するログの全てを）、/var/log/maillog に記録するように設定しています。^{※1}

/var/log/maillog を確認すると、メールの配信履歴を中心として数多くのログデータが記録されていることがわかります。

```
*.info;mail.none;authpriv.none;cron.none /var/log/messages
```

という設定がありますが、これは、ファシリティが mail と authpriv、それに cron 以外のものなら、info 以上のプライオリティのログを、/var/log/messages に記録することを意味しています。

※1 ファイルパスの先頭にマイナス記号「-」を付けることで、ファイルへの同期処理が行われなくなり、ログ出力のパフォーマンスが向上します

[練習]

1. /etc/rsyslog.conf の各行の設定を確認します。
2. /etc/rsyslog.conf の authpriv に関するログの出力先を GUI のコンソールに変更します。tty コマンドで利用している端末名を確認してください。

```
# tty
```

```
/dev/pts/1
```

↓この場合 rsyslog.conf は以下のようなになる

```
authpriv.* /dev/pts/1
```

3. rsyslog の再起動を行います。
4. /var/log/messages を確認し、rsyslog が正常に再起動したことを確認します。
5. 指定したコンソール以外のところで、useradd コマンドでユーザー logtest を作成します。この時、作成したユーザ情報が指定したコンソールに出力されることを確認します。
6. 上記で加えた変更を元に戻します。

8.2 ジョブスケジューリング

ジョブスケジューリングとは、定期的なジョブ実行や、定めた日時のジョブ実行をいいます。こういった場合に利用されるのが「cron」と「at」です。

8.2.1 cron によるジョブの定期実行

cron は定期的に処理を行うための自動実行機能を実現しています。1日に1度必ず行う処理、1年のうち特定の日のみに行う処理などを登録しておき、自動的に実行させます。

cron は `crond` というデーモンで管理されています。cron の設定は `crontab` コマンドを利用することで行います。

`crond` にジョブを登録するには、「`crontab -e`」コマンドを用います。設定ファイルが編集できる状態で開かれるので、例えば、以下のように設定を行います。編集は `vi` で行われるようになっていますので、`vi` の使用方法で作業を行います。

```
$ crontab -e
* * * * * echo "cron test" > /dev/pts/0
```

これは、毎分「`echo "cron test" > /dev/pts/0`」を実行するように登録しています。`crontab` への登録は次のような書式で行います。

分	時	日	月	曜日	コマンド
---	---	---	---	----	------

例えば、6月の毎週金曜日午前9時と10時に、コマンド `cmd` を実行したいという場合は、

0	9,10	*	6	5	cmd
---	------	---	---	---	-----

とします。分は 0~59、時は 0~23、日は 1~31、月は 1~12 として指定します。曜日は日曜日=0、月曜日=1、火曜日=2、水曜日=3、木曜日=4、金曜日=5、土曜日=6として登録します。月と曜日の指定には、Jan, Feb や Sun, Mon といった名前も使用できます。

また、特殊な設定として「*」がありますが、これは、全てを表しており、特に数値を指定しない場合に利用します。上記の例では、日付にかかわらず、他の条件を満たせば、ジョブを実行するということになります。したがって、

*	3	*	*	*	cmd
---	---	---	---	---	-----

は、午前3時台の毎分にジョブが実行されることを意味しており、

0	3	*	*	*	cmd
---	---	---	---	---	-----

は、午前3時0分にジョブが実行されることを意味しています。

また、crontab では、

```
*/10 * * * * cmd
```

という記述を行うこともできます。「/10」のように/に続けて数値を指定すると、その計算結果が整数になる値のときが実行対象として指定されたことになります。この場合、0分、10分、20分、30分、40分、50分に実行されます。

crontab に登録した設定は、「crontab -l」で確認できます。

```
$ crontab -l
* * * * * echo "cron test" >> /dev/pts/0
```

登録したジョブを一括削除するには「-r」オプションを使います。

```
$ crontab -r
$ crontab -l
no crontab for student
```

8.2.2 at によるジョブの予約

cron は定期的に繰り返し実行するジョブのスケジューリングを行うことができますが、定期的ではなく1回のみの実行でよいので登録しておきたいといった場合があります。例えば、サーバー設置場所の電源の法定点検で、ある時刻になったら自動的にシャットダウンさせたいといった場合などです。

こういった場合には、at によるジョブの実行を利用します。at は、登録した日時にジョブの実行を行うための仕組みです。cron が定期的に行うのに対して、at は1度のみ実行します。at に登録したジョブは、atd デーモンによって管理されます。

at コマンドを使うと、自動実行させたいコマンドを予約できます。at コマンドは、「at 日時」の形で実行すると「at>」というプロンプトが表示されるので、予約したいコマンドを順に設定していきます。終了するには、[Ctrl]+[D]を入力します。

時間指定の例

at 12:00	12:00 に実行
at 4:30pm	16:30 に実行
at 1:30 +3 days	3日後の01:30 に実行
at 17:00 Nov 15	11月15日の17:00 に実行

at で予約してあるジョブを表示するには、「atq」コマンドもしくは「at -l」コマンドで確認します。
また、予約してあるジョブを取り消すには、「atrm ジョブ No.」コマンドもしくは「at -d ジョブ No.」コマンドを実行します。

実行例

```
$ at 3:20
at> ls
at> <EOT>          ← [Ctrl]+[D] を入力すると<EOT>と表示されます。
job 1 at Fri Jul 31 03:20:00 2017
$ at 4:20 +1 days
at> date
at> <EOT>
job 2 at Fri Jul 31 04:20:00 2017
$ atq
1      Fri Jul 31 03:20:00 2017 a student
2      Fri Jul 31 04:20:00 2017 a student
$ atrm 2
$ at -l
1      Fri Jul 31 03:20:00 2017 a student
```

[練習]

今から1分後に、student へメールを送ります。表題は「at test」で、本文は任意です。

8.3 NTP

8.3.1 NTP サーバー

ログの記録やジョブの定時実行のため、時刻は正確でなくてはなりません。ログの記録時刻が正しくなければ、実際にいつトラブルやイベントが発生したのか追跡が難しくなります。cron や at のジョブがいつ実行されるのかわからなくなってしまいます。

そのためサーバーは NTP (Network Time Protocol) を利用して、時刻を正確に保ちます。

NTP は、ネットワーク上にある NTP サーバーと時刻を同期します。NTP サーバーは、原子時計や、GPS (Global Positioning System) 衛星などから信号を受信し、高い時刻精度を維持しています。

NTP では、通信にかかった時間を考慮して、時刻情報を設定します。これで、正確な時刻をコンピューターに設定することができます。

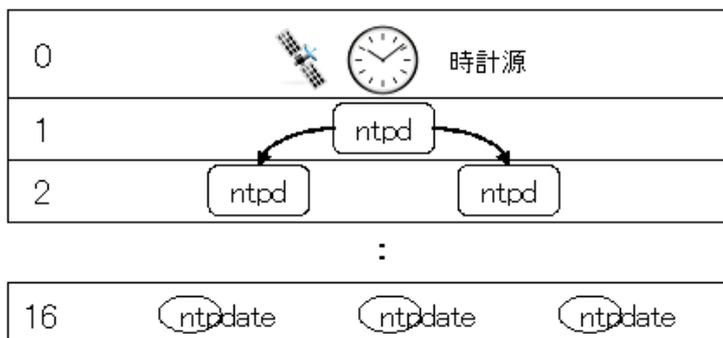
NTP サーバー群は階層構造をとることができ、原子時計や GPS を起点 (Stratum 0) とし、そこに直結しているものを Stratum 1、さらにそのサーバーを参照しているものを Stratum 2 とし最大16層まであります。Stratum 16 はクライアントのみでサーバーにはなれません。

主な公開 NTP サーバー

提供者	FQDN	Stratum
情報通信研究機構	ntp.nict.jp	1
インターネットマルチフィード(株)	ntp.jst.mfeed.ad.jp	2
国立天文台	s2csntp.miz.nao.co.jp	2
NTP POOL Project	0~3.jp.pool.ntp.org	2~4

NTP サーバーに負荷が集中しないよう、ネットワーク的に最寄りのものを選択します。また有志のサーバー群により複数の NTP サーバーを分散配置する NTP POOL プロジェクトもあります。適当なサーバーが見つからない場合は、これらを利用すると良いでしょう。

Stratum 階層



8.3.2 ntpdate

NTP サーバーにアクセスし、時刻を設定するには ntpdate コマンドを利用します。

```
ntpdate NTP サーバー ...
```

引数に NTP サーバーを指定します。この時、誤差を軽減するために複数のサーバーを指定することが推奨されています。

以下は、ntpdate コマンドを使って、NTP サーバーから時刻を取得し、システムクロックの設定を行っています。

```
# date
2017年  8月 15日  火曜日 12:08:05 JST
# ntpdate 0.jp.pool.ntp.org 1.jp.pool.ntp.org
15 Aug 12:08:19 ntpdate[17979]: adjust time server 122.215.240.51 offset -0.000147 sec
```

8.3.3 ntpd

NTP サーバーとして配信することも可能です。その場合は、/etc/ntp.conf ファイルに上位 NTP サーバーを指定し ntpd を起動します。

/etc/ntp.conf 例

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool
(http://www.pool.ntp.org/join.html).
server 0.jp.pool.ntp.org iburst
server 1.jp.pool.ntp.org iburst
server 2.jp.pool.ntp.org iburst
```

また、ntpq コマンドにより、同期の状況を表示することができます。

```
# systemctl start ntpd
# ntpq -p
```

remote	refid	st	t	when	poll	reach	delay	offset	jitter
*chobi.paina.net	131.113.192.40	2	u	5	64	1	2.222	0.116	0.105
jp.f2tec.de	10.84.87.146	2	u	5	64	1	3.289	-3.579	0.179
78.140.251.2	194.190.168.1	2	u	5	64	1	135.204	-0.011	0.179
ec2-52-198-17-9	133.243.238.244	2	u	15	64	0	4.188	-0.908	0.000

表示内容

·remote	NTP サーバー名
·refid	IP アドレスまたは時計原 (Stratum 0 の原子時計または GPS)
·st	Stratum 値
·t	通信プロトコル (l: ローカルループバック、u: ユニキャスト、m: マルチキャスト、b: ブロードキャスト)
·when	最新パケット受け取り時間([s]前)
·poll	ポーリング間隔[s]
·reach	受信結果 (直近8回)
·delay	平均遅延時間[ms]
·offset	時間の差[ms]
·jitter	時刻のばらつき[ms]

【練習】

1. ntpdate コマンドがインストールされているか確認します。もしインストールされていない場合は、ntp パッケージをインストールしてください。
2. date コマンドを実行して、システムクロックを確認します。
3. ntpdate コマンドを利用して、正確な時刻に設定します。利用する NTP サーバーは自由に選んでください。
4. 再度 date コマンドを実行し、システムクロックが変更されたことを確認します。
一度システムクロックを設定しても、コンピューターを稼働していると少しずつずれてきます。したがって、正確な時刻を維持するためには、定期的に時刻情報を取得する必要があります。ここでは、cron を利用して、毎日午後2時10分 (動作を確認できる時刻) に ntpdate コマンドを実行するようにします。
5. crontab に次の設定を追加します。
10 14 * * * /usr/sbin/ntpdate -s ntp.nict.jp

ntpdate コマンドに「-s」オプションをつけて実行すると、コマンドの実行結果を標準出力ではなく、rsyslog に出力するようになります。

8.4 オンラインマニュアル

8.4.1 オンラインマニュアルとは

Linux には、数多くのオンラインマニュアルが準備されています。コマンドの利用方法や設定ファイルの記述方法などをオンラインで調べることができます。なおオンラインマニュアルは標準化されており「見出し」なども LPIC 試験の範囲となっています。

8.4.2 オンラインマニュアルの利用

オンラインマニュアルを参照するには、man コマンドを利用します。man コマンドの引数にキーワードを指定すると、そのキーワードに対応するオンラインマニュアルが表示されます。

例: 「man df」の実行結果

```

student@h231:~
DF(1)
名前
df - ファイルシステムのディスク容量の使用状況を表示する
書式
df [options] [file...]
POSIX オプション: [-kP]
GNU オプション (簡略形式): [-ahiklmvHPT] [-t fstype] [-x fstype]
[-block-size=SIZE] [--no-sync] [--sync] [--help] [--version] [--]
説明
df は各ファイルシステムにおいて、すでに使用中のディスクの量と使用可能な
ディスクの量を表示する。
引き数がない場合、df は現在マウントされている (全てのタイプの) 全ファ
イルシステムについて使用量と使用可能量を表示する。引き数が指定されて
いる場合、df は引き数 file が含まれるファイルシステムについて表示する。
POSIX 詳細
出力はデフォルトでは 512 バイト単位であるが、-k オプションが指定された
場合は 1024 バイト単位になる。-P オプションが指定されない場合、出力フ
ォーマットは未定義である。file が通常のファイル、ディレクトリ、FIFO の
いずれでもない場合の結果は規定されていない。
GNU 詳細
引き数 file がディスクデバイスファイルで、かつそこにマウント済みのファ
イルシステムが含まれている場合、df はそのデバイスノードの属しているフ
ァイルシステムではなく、デバイスファイルに対応している方のファイルシ
ステムの使用可能量を表示する。
POSIX オプション
-k デフォルトの 512 バイト単位の代わりに 1024 バイト単位を用いる。
-P 'Filesystem N-blocks Used Available Capacity Mounted on' という
ヘッダをつけて 6 列で出力する (通常は N=512、-k オプションが指定
されたときは N=1024)。

```

マニュアルの翻訳ができていない場合は、オリジナルとなる英語のマニュアルが表示されます。また翻訳にかかる時差があるため、できるだけ英文のマニュアルを参照するよう心がけましょう。

```
$ env LANG=C man df
```

上記の df の場合、英文は 2016 年 11 月版、日本語翻訳版は 2012 年 4 月版となっています (本テキスト執筆の時点)。

8.4.3 オンラインマニュアルの構成

オンラインマニュアルには章(セクション)があり、各章には、以下のように収納される内容が決まっています。

章	解説
0	C 言語ヘッダファイル(2、3章の補足)。stdio.h、unistd.h など
1	一般コマンド。ls, cd など
2	システムコール(C 言語による OS 機能の呼び出し)。open,read など
3	ライブラリ関数(ライブラリ共通機能の呼び出し)。fopen,gets など
4	デバイスファイル。/dev/下のファイル群。tty, pts など
5	ファイル形式。/etc/passwd, fstab など
6	ゲーム、小物など
7	文字コード、プロトコルなどその他。utf8, locate など
8	システム管理コマンド。mkfs, useradd など
9	ディストリビューション依存の拡張機能(CentOS にはない)

例えば、ユーザー情報が格納されているファイル/etc/passwd およびパスワードを変更するコマンド/usr/bin/passwd はともに passwd というキーワードになります。しかし、/etc/passwd は第5章にあり、passwd コマンドは第1章にあります。

man コマンドを実行するときに、キーワードだけでなく、セクション番号を指定する場合は以下のようにします。

```
$ man 章番号 キーワード
```

passwd コマンドではなく、/etc/passwd ファイルの形式を調べる場合は次のようになります。

```
$ man 5 passwd
```

また POSIX で規定された内容には章番号のあとに「p」が、X Window System に関するものは「X」が付与されます。マニュアルの見出しも標準化されており、以下のようになります。

SECTION	見出し	解説
NAME	名前	コマンドやファイルの名前と概要
SYNOPSIS	書式	コマンドやファイルの書式・形式
DESCRIPTION	説明	コマンドやファイルの説明
OPTIONS	オプション	コマンドオプションの解説
FILES	ファイル	関連するファイル
SEE ALSO	関連項目	関連する他のマニュアル
BUGS	バグの報告	既知の問題点

マニュアルのデータは/usr/share/man/man 章番号に格納されています。各国語版は言語コードのディレクトリ下であり、日本語の場合は/usr/share/man/ja/man 章番号となります。

8.4.4 関連コマンド

man コマンドで表示されるマニュアルには長いものが多く、得てして冗長です。したがって、その内容を簡単に確認したいときのために各マニュアルの一行説明を表示させる方法として `whatis` コマンドが準備されています。

```
$ whatis crontab
crontab (1)          - 各ユーザーのための crontab ファイルを管...
crontab (5)          - cron を駆動するための一覧表
crontab (1p)         - schedule periodic background work
```

また、オンラインマニュアルを参照したいが、何をキーワードにして指定すればよいかわからない場合があります。このような場合には、`apropos` コマンドを利用するとオンラインマニュアル中の説明から検索を行うことができます。

```
$ apropos cron
cron (8)             - 予定されたコマンドを実行するデーモン...
crontab (1)          - 各ユーザーのための crontab ファイルを管...
crontab (5)          - cron を駆動するための一覧表
run-parts (4)        - configuration and scripts for running periodical jobs
anacrontab (5)       - configuration file for Anacron
anacron (8)          - runs commands periodically
crond (8)            - daemon to execute scheduled commands
crontab (1p)         - schedule periodic background work
```

上記2つのコマンド「`whatis`」「`apropos`」は共に、バイナリのデータベースを検索することで、検索時間を高速化しています。つまり、データベースを常に最新のものに更新しておくことが必要になるわけです。

このデータベースは「`mandb`」コマンド^{※2}によって更新できます。「`mandb`」コマンドは、通常ジョブスケジューリングによって毎日実行されるよう設定されています。

[練習]

1. `/etc/passwd` ファイルの GECOS 領域について、man コマンドで調べてください。
2. キーワード「`crypt`」を含むオンラインマニュアルを探してください。

※2 CentOS6 までは「`makewhatis`」コマンド

8.4.5 info コマンド

GNU プロジェクトが提供するマニュアルを参照するコマンドとして、info コマンドがあります。引数にキーワードを指定し実行します。マニュアル中に*で始まるキーワードにカーソルを合わせ、[Enter]を押すと当該部分にジャンプします。以下は「info ls」を実行した例です。

```
File: coreutils.info, Node: ls invocation, Next: dir invocation, Up: Directory listing
10.1 'ls': List directory contents
=====

The 'ls' program lists information about files (of any type, including directories). Options and file arguments can be intermixed arbitrarily, as usual.

For non-option command-line arguments that are directories, by default 'ls' lists the contents of directories, not recursively, and omitting files with names beginning with '.'. For other non-option arguments, by default 'ls' lists just the file name. If no non-option argument is specified, 'ls' operates on the current directory, acting as if it had been invoked with a single argument of '.'.

By default, the output is sorted alphabetically, according to the locale settings in effect.(1) If standard output is a terminal, the output is in columns (sorted vertically) and control characters are output as question marks; otherwise, the output is listed one per line and control characters are output as-is.
--zz-Info: (coreutils.info.gz)ls invocation, 58 lines --Top-----
Welcome to Info version 5.1. Type h for help, m for menu item.
```

Info コマンドは Emacs と呼ばれる高機能エディタを利用しており、主な操作は以下のようになります。

キー	解説
↑、↓、←、→	カーソルの移動
空白, n	次頁
p	前頁
[TAB]	ノード(分岐点)への移動、[Enter]で分岐
[Ctrl]+s	順方向へ検索(インクリメンタルサーチ)
[Ctrl]+r	逆方向へ検索
q	終了

8.4.6 その他の情報源

インターネットの検索エンジンでも同様の情報を得ることができます。検索エンジンを使うときは `manpage` というキーワードを含めるようにします。



また、Google が提供する Google Groups には、数多くの掲示板サイトのアーカイブがあり、キーワードを指定してフォーラムを検索することができます (<https://groups.google.com/forum/>)。



9 システム管理 (2)

9.1 システムの運用状況の確認

サーバーを運用する上で、ログを確認することは重要ですが、全ての情報がログから得られるとは限りません。CPU の使用率など刻々と変化するのは、管理者が主体的に情報を収集することが重要です。Linux には、サーバーを運用する上で必要となる様々なコマンドが用意されています。これらのコマンドは既に学んだものもありますが、本章では、もう一歩進んだ内容について学ぶこととします。

9.1.1 CPU 負荷の監視

uptime コマンドを利用することにより、CPU の負荷を確認することができます。不正プログラムなどが実行されると CPU に異常な負荷がかかることがあります。このような状況を迅速に発見するためには、CPU にどの程度の負荷がかかっているのか、日々把握しておくことが重要です。

```
$ uptime
22:30:12 up 1:46, 1 user, load average: 0.00, 0.01, 0.05
```

左から現在時刻、システム稼働時間、ログインしているユーザー数、CPU の平均負荷 (過去1分、5分、15分) を表示しています。平均負荷は、実行待ちプロセス数の平均値です。CPU が1つの PC では、平均負荷が1を超えていると、CPU に負荷がかかっていることを表します。

なお、w コマンドや top コマンドの表示にも uptime コマンドの実行結果が表示されます。

9.1.2 メモリ利用の確認

free コマンドを利用することにより、メモリおよびスワップ領域の利用状況を確認することができます。メモリが不足してスワップ領域が使用されるようになると、処理速度が低下してしまいます。メモリ不足を引き起こさないように注意を払う必要があります。

```
$ free
              total        used        free      shared  buff/cache   available
Mem:          1015476      82316      452192         6908      480968      764312
Swap:           839676           0       839676
```

Mem: 行には主メモリ、Swap: はスワップ領域の状況が表示されます。単位は KB ですが、-m(MB)、-g(GB)、-t(TB) で単位を切り替えることができます。-h(for Human)により自動的に補助単位を切り替えます。

各値は以下の通りです。

- ・total 総容量
- ・used 使用量
- ・free 空容量
- ・shared 共有メモリ割当量(プロセス間で共有される領域)
- ・buff/cache バッファとキャッシュ割当量(性能向上のため空容量に応じ自動割当)
- ・available 新たにプロセスを起動する時に割り当て可能な容量(実質的な空き容量)

スワップ領域の詳細は /proc/swaps に記載されています。

```
$ cat /proc/swaps
```

Filename	Type	Size	Used	Priority
/dev/sda7	partition	3071996	0	-1

スワップは通常、専用のパーティションを用意しますが、スワップが不足したなどの理由で緊急に対処したい場合は、ファイルのスワップ領域として割り当てられることも可能です。ただし、専用パーティションに比べるとパフォーマンスは低下します。

[練習]

スワップ領域がどこに割り当てられているか確認してください。

9.1.3 リアルタイムの監視

top

top コマンドを利用することにより、プロセスの状況をリアルタイムに確認することができます。デフォルトでは、CPU 使用率が高い順にプロセスが表示され、3秒ごとに新しい情報に更新されます。また、スペースキーを押すと情報が即座に更新されます。qキーを押すとコマンドが終了して、プロンプトに戻ります。

```
top - 22:46:15 up 2:02, 1 user, load average: 0.13, 0.04, 0.05
Tasks: 271 total, 2 running, 269 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.3 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 783072 total, 410552 free, 184728 used, 187792 buff/cache
KiB Swap: 1023996 total, 1023996 free, 0 used. 446460 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+  COMMAND
29026 student  20   0 130152  1928  1264 R   0.7   0.2   0:00.04 top
   811 root      20   0   7044   968   788 S   0.3   0.1   0:04.18 hypervkvpd
28008 root      20   0     0     0     0 S   0.3   0.0   0:00.18 kworker/0:0
     1 root      20   0  57156  4288  2424 S   0.0   0.5   0:00.72 systemd
     2 root      20   0     0     0     0 S   0.0   0.0   0:00.00 kthreadd
     3 root      20   0     0     0     0 S   0.0   0.0   0:00.04 ksoftirqd/0
     5 root       0 -20     0     0     0 S   0.0   0.0   0:00.00 kworker/0:0H

(省略)
```

top の主なコマンドは以下の通り。

- ・h 使い方の表示
- ・d 表示間隔の切り替え (デフォルトは3秒)
- ・u ユーザーの絞り込み (デフォルトは全員)
- ・q 終了

vmstat

vmstat コマンドを利用すると、プロセス、メモリ、スワップ、I/O、CPU などの状況をリアルタイムに確認することができます。

次の例では、5秒おきに3回分の情報を表示させています。結果の1行目は vmstat を起動してからの平

均値、2行目以降は実行した時点での状態を表示しています。単に vmstat と実行すると、1行目しか表示されないの、リアルタイムで状況を確認するにはオプションで、間隔を指定します。

```

$ vmstat 5 3
procs -----memory----- ---swap-- -----io----- -system-- -----cpu-----
 r  b  swpd   free  buff  cache   si   so   bi   bo   in   cs  us  sy  id  wa  st
 3  0     0 410940  1308 186484   0   0   21   4 1036   75  0  1 99  0  0
 0  0     0 410676  1308 186516   0   0    0   0 1016   82  0  1 98  0  0
 0  0     0 410676  1308 186516   0   0    0   0 1007   28  0  1 99  0  0

```

- procs : プロセスの概要(個)
実行待ち(r:runtime)と入出力待ち(b:blocked)のプロセス数を表します。
- memory : メモリ利用状況(KB)
swpd はスワップ領域使用量、free は空き、buff はバッファ、cache はキャッシュに割り当てられたメモリ容量を表します。
- swap : スワップ利用状況(/s)
si(スワップイン)はスワップ領域からメモリへ転送されたデータの量、so(スワップアウト)はメモリからスワップ領域へ転送されたデータの量
- io : HDD利用状況(block/s)
bi はデータの読み取り、bo は書き込み量を表します。
- system : OS の負荷(回)
in(interrupt)は割込みの回数、cs(context switch)はプロセスの切替え回数を表します。
- cpu : CPU の利用内訳(%)
us(user)ユーザー空間での利用率、sy(system)システム空間での利用率、id(idle)空き時間、wa(wait)入出力待ちの利用率、st(stolen)仮想システムの処理での利用率を表します。

9.1.4 ディスクの利用状況の確認

df

df コマンドを利用することにより、ディスクの使用率を確認することができます。「-h」オプションを指定すると、適切な単位をつけて表示してくれます。

```

$ df
ファイルシステム              1K-ブロック      使用  使用可  使用%  マウント位置
/dev/sda2                    5109760  123628 4986132    3% /
devtmpfs                     493788      0 493788    0% /dev
tmpfs                        508240     144 508096    1% /dev/shm
tmpfs                        508240    7072 501168    2% /run
tmpfs                        508240      0 508240    0% /sys/fs/cgroup
/dev/sda3                    5109760 3779436 1330324   74% /usr
/dev/sda6                    508580   25916 482664    6% /home

```

(省略)

-i オプションを指定すると、iノードの使用率が表示されます。iノードが不足するとファイルの新規作成ができなくなるので、容量だけでなく、iノードにも注意しましょう。

```
$ df -i
ファイルシステム          Iノード   I使用   I残り I使用% マウント位置
/dev/sda2                 2560000  4024 2555974   1% /
devtmpfs                  123447   359 123088   1% /dev
tmpfs                     127060    7 127053   1% /dev/shm
tmpfs                     127060   490 126570   1% /run
tmpfs                     127060   16 127044   1% /sys/fs/cgroup
/dev/sda3                 2560000 121842 2438158   5% /usr
/dev/sda6                 256000   26 255974   1% /home
(省略)
```

du

du コマンドを利用すると、ディレクトリやファイルの容量を確認することができます。

```
$ du -h public_html/
4.0K    public_html/
```

ファイル名表示オプション(-a)

```
$ du -ah public_html/
4.0K    public_html/index.html
4.0K    public_html/
```

df コマンドの結果、ディスク使用率が高くなっている場合は、du コマンドを利用して、どのファイルがディスクを圧迫しているのかを確認します。その後、ディスクを増設したり、ファイルを削除したりするなど、適切な対策を行っていきます。

/home ディレクトリや/var ディレクトリには、頻繁に更新されるデータが入るため、ディスクの使用率には特に注意が必要です。用途に応じて十分な容量を用意しておくとともに、独立したパーティションに配置してメンテナンス性を高めておくことが望ましいでしょう。

[練習]

1. /home の空き容量と、i-node 残量(ファイル数)を調べてください。
2. ホームディレクトリの容量を調べてください。

9.2 冗長性のあるシステム構成の構築

コンピューターは、どんなに良質な部品を使っても故障する可能性があります。故障による、サービス停止、データ消失といった重大な事故に備える方法に冗長化があります。冗長化により、一部に障害が発生した場合にも継続してシステムを動作させることが可能になります。

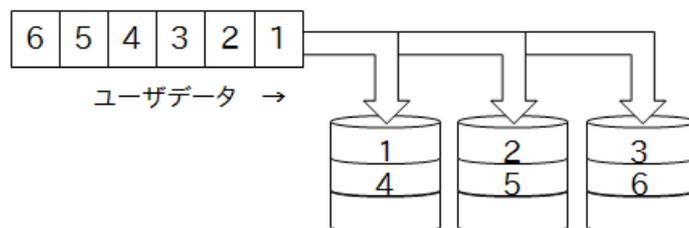
9.2.1 RAID

複数のハードディスクを組み合わせることで1台の大容量ハードディスクのように扱う技術を RAID (Redundant Arrays of Independent / Inexpensive Disks) と呼びます。RAID によりディスクの冗長化や、大容量化、高速化を行うことが可能です。

RAID には、専用のハードウェア (RAID コントローラ) を使用する「ハードウェア RAID」と、OS の機能で実現する「ソフトウェア RAID」があります。各ディスクの組み合わせ方をレベルと呼び、速度や信頼性によって RAID 0 から RAID 6 までの 7 種のレベルが存在します。ほとんどの場合、実際に利用されるのは、RAID 0、RAID 1、RAID 5、RAID 6 の 4 種です。Linux カーネルは、この 4 種のソフトウェア RAID をサポートしています。

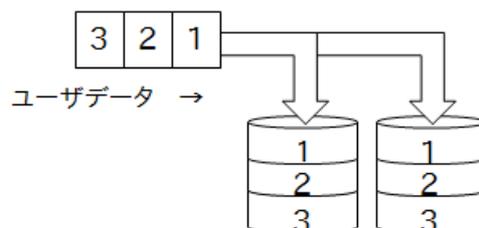
RAID 0 (ストライピング)

書き込み・読み出しを2台以上のディスクに分散して処理します。これにより、複数のディスクに負荷分散が可能となり、高速に読み書きが行えます。複数のディスクからなる大容量の RAID が構成できます。しかし、冗長性がなく、どれか1台のディスクが故障することで、RAID 全体が読み出し不能となる欠点があります。



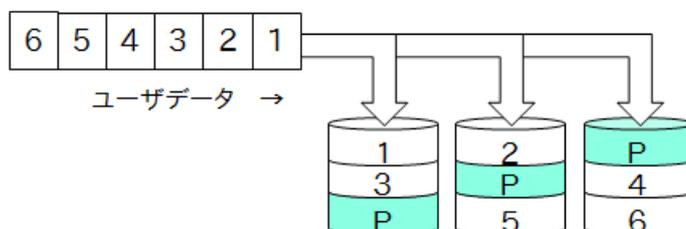
RAID 1 (ミラーリング)

同一のデータを2台以上の複数のディスクに書き込みます。1台のディスクが故障しても、他のディスクで処理を続行できるため、冗長性に優れています。しかし、同じデータを台数分のディスクに対して書き込むため、ディスクの使用効率は1/台数分になりコストがかさみます。



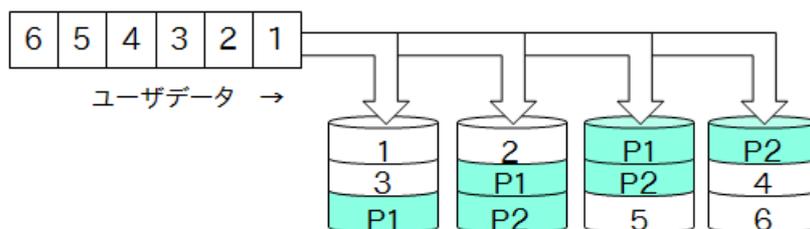
RAID 5

データとそのデータから生成されたパリティ情報を3台以上の複数ディスクに分散して書き込みます。万一ディスクの1台が故障しても、残ったデータとパリティ情報から、失われたデータの復元ができ、冗長性があります。パリティ情報に必要な容量はディスク1台分で済むため、ミラーリングよりも効率よく冗長化と、負荷分散による高速な読み書きも可能です。ただし、パリティ処理がオーバーヘッドとなり、RAID 0 ほどのスピードは確保できません。



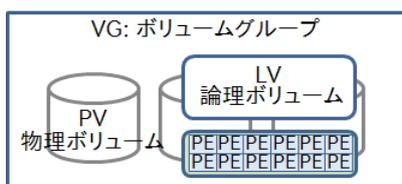
RAID 6

動作原理そのものは RAID 5 とあまり変わりありません。ただし、データとそのデータから生成されたパリティ情報2種類を、4台以上のディスクに分散して書き込みます。RAID 5 は冗長性が確保されていますが、同時に2台のディスクの故障には対応できません。これに対して、RAID 6 は2種類のパリティを分散して保持することにより、同時に2台のディスクの故障に対してもデータの復元が可能です。



9.2.2 LVM

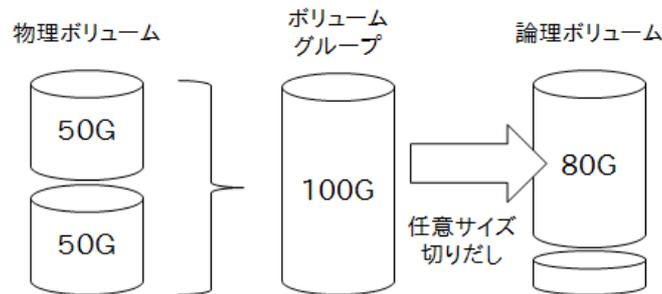
LVM (Logical Volume Manager) は、固定的なパーティションの制限超え、柔軟なディスク管理を実現する技術です。複数のハードディスクを「ボリュームグループ」としてまとめ、そこからパーティションに相当する「論理ボリューム」を切り出して使用します。



- PV (Physical Volume) ハードディスクドライブやパーティション
- VG (Volume Group) PV の集合
- LV (Logical Volume) VG から切出した仮想的なパーティション
- PE (Physical Extent) 仮想的なブロックサイズ (LV の構成単位)

通常、ディスクのパーティションは、システム導入時に必要な容量を割り当てるため、後からサイズ変更を行うことはできません。そのため、容量が不足した場合などは、別途パーティションを用意し、既存のデータを移動(データの引越)する必要があります。また、データの引越には、複数回のシステムの再起動を行う必要があります。

LVM は後から論理ボリュームのサイズ変更がシステムの再起動なしで可能となります。またボリュームグループへのディスク追加や、複数のハードディスクにまたがる論理ボリュームを構成することも可能です。



LVM によって柔軟なディスク管理が可能となりますが、冗長化を確保することはできません。前述の RAID と LVM を組み合わせることによって、冗長性を確保しつつ、柔軟なディスク管理を行うことが可能になります。

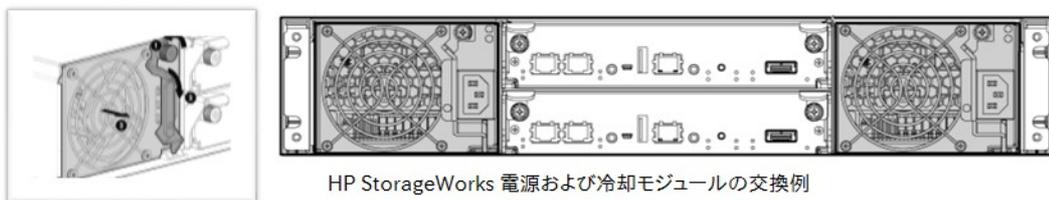
[練習]

pvs、vgs、lvs コマンドを使って現状の確認をしてください。

9.2.3 冷却ファンや電源の二重化

コンピューターの中で壊れやすい(消耗しやすい)部品として、まずハードディスクが挙げられますが、他にも冷却ファンや電源などの可動部品も、同様に消耗しやすい部品といえるでしょう。

万一に備え、冷却ファンや電源を二重化しているサーバーも数多くありますし、1台の冷却ファンや電源だけに依存しないような構造になっているサーバーも少なくありません。特にハードディスクや電源の中には、稼働状態のまま故障部品の交換が可能な技術(ホットスワップ)を採用しているものもあります。



HP StorageWorks 電源および冷却モジュールの交換例

他にも、CPU やメモリ、バスといった全ての構成要素を二重化したコンピューターも存在します。

9.3 バックアップ

RAID によって冗長性を確保することはできますが、万が一の事態に備えて、バックアップは必要です。想定されるケースには次のようなものがあります。

- 操作ミスによるデータの消失
- ウィルスや不正侵入によるデータの消失
- 自然災害による機器の損傷

また、大掛かりなシステム変更前にも、移行時のトラブルによるデータの消失などに備え、バックアップを行うのが適当とされています。

9.3.1 バックアップの対象

ハードディスクの大容量化によってバックアップを取るデータ量が増えると、バックアップメディアに必要な容量や、バックアップ処理に要する時間も増大します。

データには頻繁に更新されるものとそうでないものがあるので、毎回全てのデータをバックアップするのは効率的ではありませんし、コストも考慮しなくてはなりません。そこで、更新が頻繁に行われるデータこそ、頻繁にバックアップする必要があるということになります。

ディレクトリ	バックアップの必要性
/etc	各種設定ファイルが格納されているため
/home	各ユーザーのホームディレクトリが可能されているため
/var	ログ、メールなど常に情報が蓄積されているため

これに対して、インストール時に復元できる(インストールし直せる)データは、バックアップの必要性は低いと言えます。/bin や/sbin、/dev などのディレクトリがこれに該当します。

9.3.2 バックアップの種類

バックアップには、次に説明するように、いくつかの種類があります。これらを使い分けることで、バックアップのサイズやバックアップ処理に要する時間、リストア(バックアップからの復旧)作業にかかる手間のバランスを取ることができます。

完全バックアップ(フル・バックアップ)

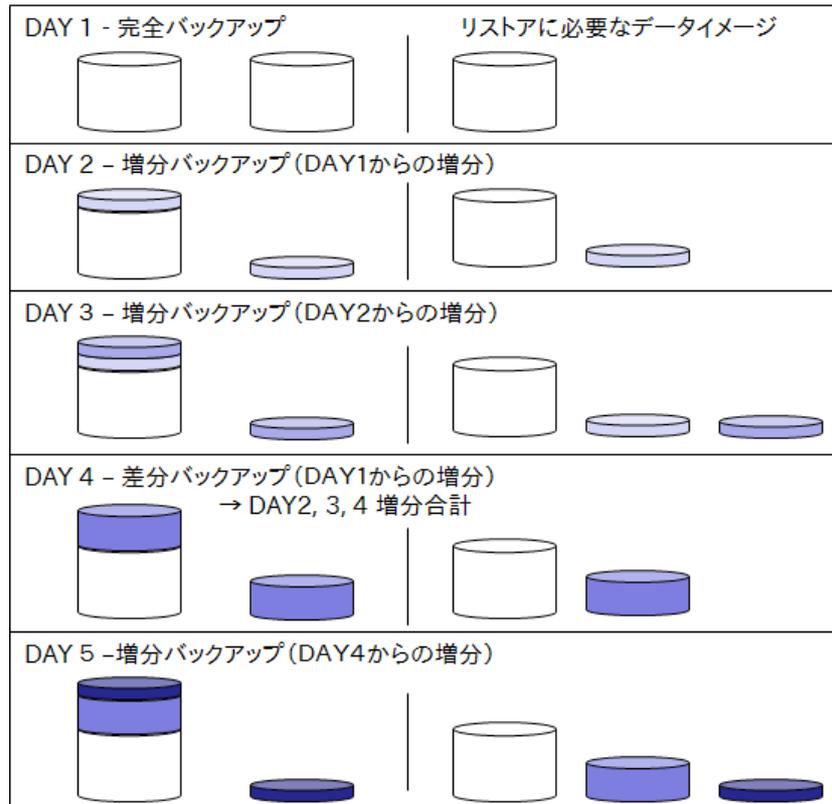
システム上の全てのファイル、もしくは指定したパーティション内の全てのファイルをバックアップします。バックアップのサイズが大きくなり、バックアップ処理に時間もかかりますが、リストアは1回の作業で済みます。

差分バックアップ

前回の完全バックアップからの変更分がバックアップされます。したがって、バックアップのサイズや処理にかかる時間が短縮されます。ただし、最後の完全バックアップからの時間が経過するほどバックアップのサイズも大きくなります。リストアには、完全バックアップと最新の差分バックアップが必要になります。

増分バックアップ

前回のバックアップ(完全バックアップ、差分バックアップ、増分バックアップ)からの変更点をバックアップします。バックアップ処理は短時間で済みますが、リストアには、完全バックアップと差分バックアップ、それ以降の全ての増分バックアップが必要になります。



容量が大きい場合は完全バックアップ、巨大なデータの場合は完全バックアップ後に定期的な増分バックアップを取得することがよく行われています。

9.3.3 XFS でのバックアップとリストア

バックアップ

XFS でのバックアップは `xfsdump` コマンドにより行います。`ext2/3/4` は `dump` を使用します。

```
xfsdump オプション バックアップ対象ファイルシステム
```

主なオプション

オプションと引数	解説
<code>-l</code> ダンプレベル	ダンプレベルは0~9があり、0がフル・バックアップ、1~9は前回取得してからの差分を表す。例えば日曜日に0で取得し、月曜に1を指定した場合は日曜からの増分を指す。更に2を指定すると、月曜からの増分となる。
<code>-f</code> 出力ファイル	ダンプしたデータを出力する先を指定。
<code>-L</code> セッションラベル	この操作のコメント(セッションラベル)を指定。指定がないと入力要求がある。
<code>-M</code> メディアラベル	テープなどのメディア管理用名称。指定がないと入力要求がある。

実行例

```
# xfsdump -l 0 -L "boot backup" -M "disk" -f /var/boot.bkup /dev/sda1
xfsdump: using file dump (drive_simple) strategy
```

```

xfsdump: version 3.1.4 (dump format 3.0) - type ^C for status and control
    (中略)
xfsdump: dump size (non-dir files) : 140959472 bytes
xfsdump: dump complete: 1 seconds elapsed
xfsdump: Dump Summary:
xfsdump:  stream 0 /var/boot.bkup OK (success)
xfsdump: Dump Status: SUCCESS

```

最後が SUCCESS(成功)で終わっていることを確認しましょう。

リストア

リストアは xfsrestore で行います。dump 同様 ext2/3/4 では restore が用いられます。

```
xfsrestore オプション 展開するディレクトリ
```

主なオプション

オプションと引数	解説
-i	対話モードでの実行。
-f 出力ファイル	読み込むファイル(xfsdump の出力ファイル)。

先のファイルをカレントディレクトリへリストアする例

```

# xfsrestore -i -f /var/boot.bkup .
xfsrestore: using file dump (drive_simple) strategy
    (中略)
xfsrestore: directory post-processing
===== subtree selection dialog =====
the following commands are available:
    (中略)
-> ls
          202557 initramfs-3.10.0-514.el7.x86_64kdump.img
          74  initramfs-3.10.0-514.el7.x86_64.img
          77  initrd-plymouth.img
          :
-> add initrd-plymouth.img
-> extract
----- end dialog -----
    (省略)
xfsrestore:  stream 0 /var/boot.bkup OK (success)
xfsrestore: Restore Status: SUCCESS

```

【練習】

1. /boot のフルバックアップを xfsdump で取得し、/var/boot.bkup を作成。

2. /boot に/etc/hosts をコピーし、ダンプレベル1で再びバックアップを取得し、
/var/boot1.bkup を作成。この時のファイル容量を比較する。
3. /root に移動し、xfsrestore を使ってダンプの先の内容を確認、任意のファイルをリストアする。

10 セキュリティ入門

10.1 セキュリティとは

コンピューターセキュリティの定義は様々ですが、一言で言えば、「安心してコンピューターが利用できるようにその環境を保つこと」です。日頃何気なく利用しているコンピューターも、何らかの原因で突然停止してしまったり、見知らぬ誰かに勝手に使用されてしまったりする可能性があります。したがって、日常利用している環境を維持するためには、気配りが必要になります。コンピューターを安定して利用できるよう工夫を施すことを「セキュリティを確保する」といいます。

10.1.1 守るべきもの

セキュリティを確保するためにはまず、何が大事なのかを明確にする必要があります。守るべきもののはっきりしないことには、どう守っていけば良いのか、方針を確定できません。したがって、守るべきものをしっかりと見据えて、セキュリティに対する企業の指針（セキュリティポリシー）を明文化することが必要になってきます。おおよそ考えられる項目は以下の3点です。

第一にハードウェアを守ること。もし、サーバー機器が盗難にあったらサービスは提供できません。サーバー機器は高価なものが多いので盗難に対する警戒が必要です。

第二にサービスの提供そのものを守ること。提供していたサービスが止まったままでは、その企業は社会的な信用を失いかねません。一度失った信用を回復するのは容易なことではありません。場合によっては企業の存続に関わってきます。

第三にデータを守ること。たとえ盗難にあったとしてもサーバー機器は再度入手可能です。しかし、一度失ったデータを元に戻すことはできません。バックアップの必要性はここにあります。バックアップがあれば、リストアすることが可能になります。

10.1.2 セキュリティを犯すもの

セキュリティが確保できない理由は大きく分けて以下の3つの要因が考えられます。

第一に利用者による操作ミスなど、意図せず起こるトラブルによるもの。特に管理者権限での作業中に大切なファイルを誤って削除したり、重要なプロセスを停止させてしまったりといったことが考えられます。作業手順を明文化し、確認しながら作業することが必要です。

第二に外部からの攻撃者によるもの。ソーシャルエンジニアリングによるパスワードクラック、不正アクセス、踏み台など、様々な状況が想定されます。内部関係者による不正操作もこの範疇に入ります。

第三に自然発生的な災害によるもの。地震、火災、水害など様々な災害が考えられます。

10.1.3 セキュリティ向上の方法

前述の様々な問題に対処して、セキュリティを確保するにはどうしたら良いでしょう。

まず、基本的な考え方は「起こりうる問題を可能な限り排除する」ことです。極論を言えばサーバーやサービスを運用していなければ、こういった問題は起こりえません。つまり、様々なサービスや機能を利用しているからこそ、それに伴う問題を抱えることにつながるわけです。利用するサービスや機能を必要最小限にすることで、起こりうる問題も最小限に食い止めることができます。

そして、「起こりうる問題を可能な限り明確化し、対策を講じる」ことが必要です。その問題1つ1つに対して、できる限りの対策を講じます。また、問題の重要度による切り分けも必要になってきます。さらにこういった問題には完璧な対策がない場合も多く、これに対処するために、二重三重の対策を講じておくことも必要になってきます。こういった作業の積み重ねでシステム全体のセキュリティを向上させることができます。

10.2 セキュリティ対策

10.2.1 物理構成に対するセキュリティ

ハードウェアへのリスクとしては物理的な破損だけでなく、空調の不備によるシステムのオーバーヒート、粉塵による可動装置の劣化、電磁波によるデータ消失、不安定な電源といった外部環境要因も数多く存在します。多くのサーバーは天災・人災に伴う上記のリスクを回避するため、データセンターと呼ばれる専用の施設内に設置されています。データセンターは外界と遮断されており、教育・訓練を受けた専任スタッフが運用にあたっています。予め決定された以外の操作は行えず、利用者であっても簡単に入場できない仕組みになっています。

特に近年では地震などの天災に備えるため地理的に離れた場所にバックアップセンターを設けるなど、データセンターの需要が伸びています。

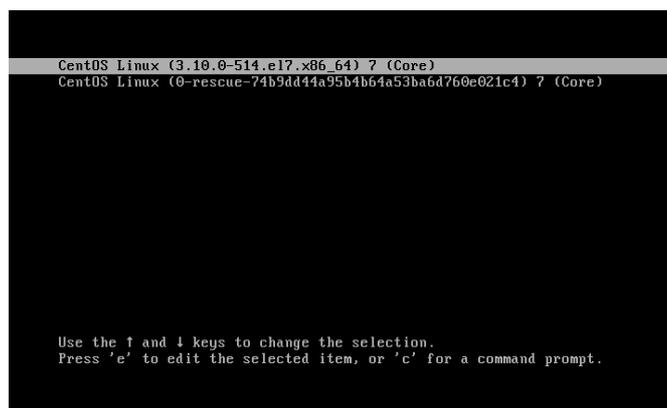
10.2.2 コンソールの横取り

直接サーバーを操作できる環境であれば、電源やリセットボタンを押して無理やりシステムを停止することが可能です。その後、何らかの細工をしてシステム管理権限を奪取することを「コンソールの横取り」といいます。

Bootメニューが表示されている時に、カーネルを選び[e]を押すと、起動パラメータを修正することができます。これを悪用し、通常のレスキューモードではなく、パスワードなしのシェルを起動するよう細工することができます。この方法によれば、管理者パスワードを知らなくとも、UID=0の隠しユーザーを作成し、そのシステムを乗っ取ることが可能です。

認証のないレスキューモード

カーネル選択画面で、使用するカーネルをカーソルキー(↑↓)で選択し、[e]を押すと、起動パラメータの編集ができます。



```
CentOS Linux (3.10.0-514.el7.x86_64) 7 (Core)
CentOS Linux (0-rescue-74b9dd44a95b4b64a53ba6d760e021c4) 7 (Core)

Use the ↑ and ↓ keys to change the selection.
Press 'e' to edit the selected item, or 'c' for a command prompt.
```

編集画面が表示されたら、linux16 で始まる行を探し、末尾に「rw init=/bin/sh」を追加します。

```

insmod part_msdos
insmod xfs
set root='hd0,msdos1'
if [ x${feature_platform_search_hint} = xy ] then
    search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1' 72190c8c-d457-42cb-9c7f-ca54f1c685eb
else
    search --no-floppy --fs-uuid --set=root 72190c8c-d457-42cb-9c7f-ca54f1c685eb
fi
linux16 /vmlinuz-3.10.0-514.el7.x86_64 root=UUID=6b424327-1b53-46da-8c72-1650faf93991 ro crashkernel=auto rhgb quiet LANG=ja_JP.UTF-8
initrd16 /initramfs-3.10.0-514.el7.x86_64.img

Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
discard edits and return to the menu. Pressing Tab lists
possible completions.

```

コンソールではキーボード配列が異なるため、[=]は[^]に割り当てられています。

[Ctrl]+[x]を押すと、変更した内容でOSが起動します。その結果、パスワード入力なしでいきなりシェルが起動します。確認できたら、init プロセスを起動します。

```

sh-4.2# (いきなり root で操作が可能)
sh-4.2# exec /sbin/init          ← (通常のOS起動を継続)

```

Boot メニューのパスワード保護

上記の不正操作を回避するために、Boot メニューにパスワード認証を追加します。

```
grub2-setpassword
```

この内容は /etc/grub2/user.cfg に記録され、削除するとパスワード認証は無効となります。ユーザー名は root となります。

【練習】

1. 講師による手順に従い、レスキューモード(認証なし)で再起動します。
2. root でログインできることを確認します。
3. マルチユーザーモードへ移行します。
4. grub2-setpassword でコンソールパスワードを設定します。
5. OSを再起動し Boot メニューを操作します。パスワード入力が行われることを確認します。
6. 確認できたら通常のOSを起動し、GRUB2の設定を元に戻しておきます。

10.2.3 ネットワークからの侵入

セキュリティ上の問題で最も深刻なのがネットワークを経由した不正アクセスです。この場合、特定のサーバーに侵入することを目的とした、高度なスキルの持ち主(クラッカー)による場合と、悪意を持った素人(スクリプトキディ)による場合の2種類に分けることができます。前者は、まだ誰も知らないようなソフトウェアのセキュリティホールを自ら見つけ出し、それを衝くような攻撃手法を考えだすことができるようなスキルの持ち主です。前者は非常に恐ろしい存在ですが、数多くは存在しませんし、利害目的が比較的是っきりしていますからの絞って対処できる場合もあります。しかし後者は、インターネット上で公開されている、暴き出されたセキュリティホールを衝くためのさまざまなツールを利用して、手当たり次第に攻撃してくる愉快犯がほとんどです。スキルは高くありませんが、数が多く繰り返し攻撃されることがあります。

攻撃は主に次のような手順を踏んで行われます。

1. サーバーの選定
2. 提供されるサービスの確認(ポートスキャン)
3. 脆弱性(セキュリティホール)の確認
4. 攻撃・侵入

前述のような理由から、スクリプトキディによる攻撃は、ほとんどが既知のセキュリティホールなどに対する攻撃です。

10.2.4 ソフトウェアのアップデート

ソフトウェアでは、バグやセキュリティホールが発見されることがあります。こういった情報はインターネットを通じて逐次確認していくことができます。

セキュリティに関する情報源例

- ・JPCERT - <https://www.jpccert.or.jp/>
- ・JPA - <http://www.ipa.go.jp/>

もちろん、問題のあるソフトウェアをそのまま使用し続けるのは、セキュリティ上好ましくありません。特に、セキュリティホールが発見された場合、そういった「穴」を衝く攻撃方法は、瞬く間にインターネット上で拡散され、攻撃用ツールなども入手可能となるためです。

OSS の場合、最初の対応策として、ソースコードに当てるパッチ(patch)が、次にパッチ適応済みのtarball が公開されます。最後に各ディストリビュータからアップデート情報が提供されることになり、タイムラグが発生することがしばしばあります。

したがって、最低限の対処方法として、常に最新のアップデート情報に気をつけ、必要に応じてパッケージのアップデートを心がけるようにしましょう。

【練習】

root ユーザにて yum update コマンドを実行して、パッケージを最新のものにします。

10.2.5 ソフトウェアの稼働状況

Linux では数多くのパッケージが提供されていますが、むやみやたらに導入すると思わぬセキュリティホールに遭遇するといったリスクが増大します。不必要なパッケージは導入しない、稼働させないことが基本です。例えば TELNET サーバーは必要でしょうか、SSH で代替できないでしょうか。

FTP、NFS、Samba 等のファイル共有サービスを本当にインターネット上で公開する必要がありますか。

このように本当に必要なサービスなのか、公開する必要があるのか、サービスを提供することで発生するリスクは何かを十分検討する必要があります。

また OSS では完成度が低いものの、広く公開することで機能の充足や安定性を高めるといった手法が用いられます。新しいソフトウェアはこのようなリスクを含むことを分かった上で利用しなければなりません。

またシステム管理者として意図的に起動していないサービスであっても、デフォルトで起動されているようなものも存在します。管理者が意識をしていないサービスの脆弱性は発見や対策が後手に回ることがあります。必ず自動起動されるサービスが何かを把握し、不要であれば停止・アンインストールといった処置が必要です。

そこで、まず起動後のサービスの稼働状態を、`ps aux` コマンドで確認します。意図しないプロセスが起動しているようであれば、調べた上で停止するようにします。`inetd` などのスーパーサーバー経由で起動するものも気をつけましょう。これらのサービスはクライアントからの要求が来ない限り、プロセスとしては常駐しないため、確認できないことがあります。ただ、この場合 `inetd` によって、ポートの監視は行われていますから、`netstat` コマンドで Listen ポートのチェックを確実にしておきます。

起動時の稼働サービスの設定状態は「`systemctl list-unit-files`」コマンドで確認します。

【練習】

1. `ps aux` コマンドや `netstat -antup` コマンドで稼働中のサービスを特定します。
2. 不必要なサービスがあれば、適宜停止させます。
3. `systemctl list-unit-files` コマンドで起動時の起動対象をチェックします。
4. 起動する必要のないサービスを、`systemctl disable` コマンドでシステム起動時の対象から外します。
5. システムを再起動し、もう一度稼働中のサービスを確認します。

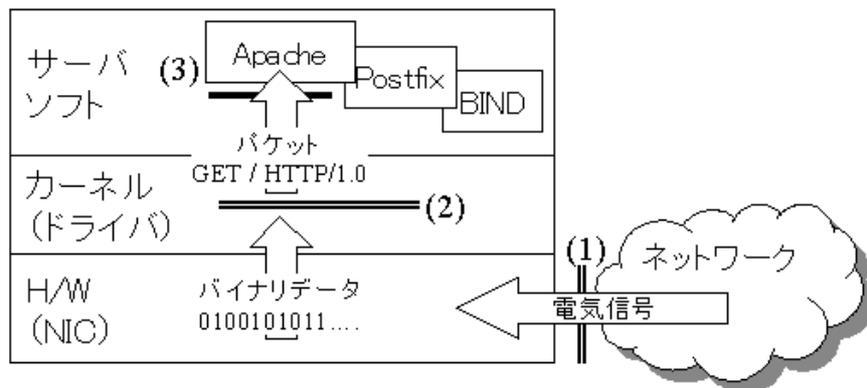
開いているポートがどのサービスによるものなのか判断がつかない場合があります。この場合、「`lsof -i:ポート番号`」コマンドを実行して、動作中のプログラムの PID を調べ、場合によっては kill します。

10.3 パケットフィルタリング

10.3.1 パケットフィルタリングとは

ネットワークを行き交うデータは「パケット」と呼ばれる細かな塊に分割されます。データを受信するコンピュータは、ネットワークケーブルからネットワークインターフェイスカード(NIC)を通してデータを受け取ります。

NICは届いた電気信号を数値データに変換し、ネットワークドライバに渡します。ネットワークドライバは数値データを文字などに変換し、各サーバーソフトへ渡します。このようにパケットがApacheやBINDなどのサーバーソフトに届くまでには、図のような流れをたどっています。



ネットワークを介するサービスにおいて、セキュリティを確保するためには、そのサービスにアクセスできるコンピュータやユーザーを制限する必要があります。

アクセス制限を行う場所は、大きく以下の3か所となります。

- (1) ネットワークに入ってくるところで制限する
- (2) ネットワークドライバで制限する
- (3) サーバーソフトで制限する

この中で、個々のサーバーソフト自身によるアクセス制限は、折に触れ学んできました。上図(3)の位置で制限を行います。一般的には、サーバーソフトの設定ファイルにアクセスを許可(又は拒否)するIPアドレスやドメインなどを記述し設定します。例えば、SSHサーバーでは、`/etc/hosts.allow`と`/etc/hosts.deny`ファイルに記述しました。また、Apache HTTPサーバーも、`Order`、`Allow`、`Deny`ディレクティブによってアクセス制限を行いました。

これ以外に、サーバーコンピュータに届く以前、つまりNICに入ってくる前(ルータやファイアウォールなどで)、パケットをフィルタリングしてしまう方法があります。この方法は、上図(1)でアクセス制限を行うこととなります。

今回は上図(2)に相当し、カーネルのTCP/IP処理内で行われます。パケットの送信元、宛先、ポート番号、プロトコルなどの情報を元にパケットの送受信を制限する方法です。その条件を満たしていないパケットは丸ごと破棄してしまいます。

このことから、この方法は「パケットフィルタリング」と呼ばれます。

パケットフィルタリングのツールとして、Netfilter が採用されており、Linux カーネル 2.4 系以降では、Netfilter の設定に、iptables と firewalld が利用できます。今回は iptables コマンドを利用します。

10.3.2 パケットフィルタリングの前提

パケットフィルタリングを行うには、まずどのようなサービスを提供し、その為にはどのような条件でパケットを受け入れるかを把握しておく必要があります。

主要な確認項目は以下のようにまとめられます。

- どのようなサービスを提供するのか (WWW、メールなど)?
- サービスごとにどのホストからのアクセスを許可するのか?
- サービスが使用するプロトコルに応じて、どのパケットを通過させるか?

次節で詳しく解説する iptables コマンドは、宛先、送信元、プロトコル、ポート番号に応じたパケットフィルタリングの設定を行うことができます。このため、各サービスがどのポート番号を用いているかをきちんと理解している必要があります。

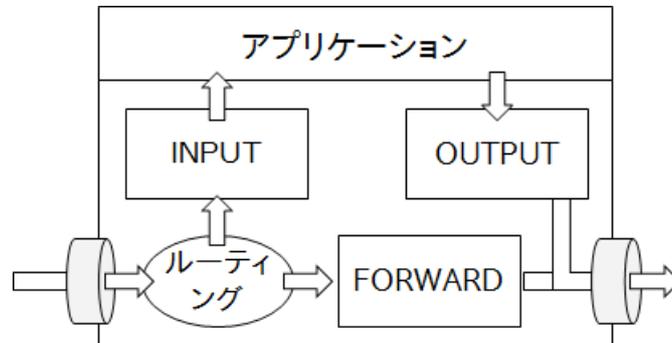
代表的なサービスとプロトコル、ポート番号

サービス名	プロトコル	ポート番号
ftp-data, ftp	TCP	20,21
ssh	TCP	22
telnet	TCP	23
SMTP	TCP	25
domain	UDP, TCP	53 (UDP: 名前解決、TCP: ゾーン転送)
HTTP	TCP	80
POP3	TCP	110
NTP	UDP	123
IMAP4	TCP	143
HTTPS	TCP	443

10.4 iptables コマンド

10.4.1 パケットの経路

ネットワークから受け取ったパケットは、次のような手順で処理されます。



iptables では、この INPUT、OUTPUT、FORWARD という3つの経路に対して、それぞれ通過させるパケットのルールを設定できます。この経路を「チェーン」と呼び、ポリシー（基本方針）とルールを付け加えてフィルタリングの設定を行います。

10.4.2 iptables の開始

yum コマンドで iptables-services をインストールします。iptables は firewalld と競合するため、後者の自動起動は無効にしておきます。また OS 起動時より iptables を有効にさせるため、systemctl コマンドで enable 設定にしておきます。

iptables のインストール

```
# yum install iptables-services
```

firewalld の無効化（既に設定済み）

```
# systemctl stop firewalld
# systemctl disable firewalld
```

iptables の有効化

```
# systemctl enable iptables
# systemctl start iptables
```

「iptables -L」は、現在設定されているポリシーとルールの一覧が表示されます。^{※1}

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

これら3つのチェイン INPUT、OUTPUT、FORWARD に対して、それぞれのパケットフィルタのルールを付け加えます。それぞれのルールは以下のように表示されます。

```
target     prot opt source                destination
```

左から、ターゲット、プロトコル、オプション、送信元、宛先です。

ターゲットとは、条件にマッチするパケットをどうするかという、処理の行き先です。既に定義されている特別なターゲットには、ACCEPT、DROP などが挙げられます。

ターゲット	解説
ACCEPT	当該パケットを通過させる
DROP	当該パケットを廃棄(エラーは通知しない)する
REJECT	該当パケットを遮断しエラーを通知する
LOG	ログへ記録する

ターゲットに続くプロトコル、オプション、送信元、宛先などは、フィルタリングするパケットの条件を表す部分です。ここでのプロトコルとは、HTTP や FTP といったアプリケーション層のプロトコルではなく、TCP、UDP、ICMP などの下位層のプロトコルを指定します。送信元、宛先の部分では、対象のホストおよびポート番号を指定します。

※1 ルールに何か設定されている場合は、「iptables -F」にてルールを削除してください

10.4.3 フィルタリングの方針

実際の設定を行う前に、フィルタリングの方針 (Policy) を確認しておきましょう。まず、今回提供するサービスは Web サービスとネームサービス、そしてメールサービスということにしておきます。また、リモートアクセスに SSH を利用することとします。

一般にパケットの通る 3 つの経路に関して、

- INPUT は基本的に全て拒否 (必要のあるポートのみ開ける)
- FORWARD は基本的に全て拒否 (必要のあるポートのみ開ける)
- OUTPUT は基本的に全て許可

という方針がとられます。システムに入ってくるパケットや、別のホストに中継するパケットに関しては、厳しいチェックを行う必要があるからです。このときの「基本的に許可 (または拒否)」という方針をポリシーといいます。

先の「iptables -L」を実行した際に表示される結果を見てみましょう。

```
Chain INPUT (policy ACCEPT)
```

この例では、「policy ACCEPT」という記述がされています。これはルールが指定されていないパケットに関しては、全て通過させるという意味を指しています。

フィルタリングポリシーは各チェーンにおいて、ルールに記述されていないパケットに対するターゲットを指定するものです。先ほどの方針と照らし合わせれば、以下のようになります。

チェーン	ターゲット
INPUT	DROP
FORWARD	DROP
OUTPUT	ACCEPT

チェーンのフィルタリングポリシーを変更するためには、以下のようなコマンドを実行します。

```
# iptables -P INPUT DROP
```

「-P」はポリシー設定を、「INPUT」はチェーン、「DROP」はターゲットになります。つまり、INPUT チェインの基本ポリシーは、ターゲット DROP に送られるよう設定しています。

これで基本的には、(ルールの適用されない) 全てのパケットは、アプリケーション側に到達する前に破棄されることとなります。

【練習】

1. 上記の例にならって、INPUT チェインのポリシーを DROP に設定します。
2. FORWARD チェインのポリシーも DROP に設定します。
3. 全てのルールを削除します。
この状態で全ての外部のネットワークからの通信が不可能になっていることを確認します。
Web ブラウザで「<http://www.google.com>」にアクセスしてみます。
4. ping コマンドで www.google.com に対して、疎通できるか試してみます。
5. 全ての受信が拒否されていることを確認します。
6. INPUT チェインを ACCEPT に戻します。

10.4.4 パケットフィルタリングの設定

次に、提供するサービスと利用するサービスを確認して、これに対する適切な設定を行う必要があります。本章では、以下のサービスを提供するものとします。

プログラム	プロトコル:ポート番号
Apache HTTP サーバー	TCP:80
BIND ネームサーバー	TCP/UDP:53
Postfix メールサーバー	TCP:25
Dovecot POP サーバー	TCP:110
SSH サーバー	TCP:22

したがって、以下の条件を満たすパケットは通過させる必要があります。宛先アドレスには 10.20.142.6 を例に挙げています。

プロトコル	送信元アドレス	宛先アドレス	送信元ポート	宛先ポート
TCP	Any	10.20.142.6	Any	80
TCP	Any	10.20.142.6	Any	53
UDP	Any	10.20.142.6	Any	53
TCP	Any	10.20.142.6	Any	25
TCP	Any	10.20.142.6	Any	110
TCP	Any	10.20.142.6	Any	22

これ以外にも、ローカルホスト自身がクライアントとして外部のネームサーバーと通信できなければ、様々な問題が起こります。また Web やメールなどのアクセスも必要になります。したがって、以下のポートを開く必要もあるでしょう。

プロトコル	送信元アドレス	宛先アドレス	送信元ポート	宛先ポート
TCP	Any	10.20.142.6	80	Any
TCP	Any	10.20.142.6	53	Any
UDP	Any	10.20.142.6	53	Any
TCP	Any	10.20.142.6	25	Any
TCP	Any	10.20.142.6	110	Any
TCP	Any	10.20.142.6	22	Any

また、ローカルホストとの通信は全て許可して問題ないので、ループバックアドレスから送られてきたパケットは許可します。また ping などが用いる ICMP に関しては基本的には通過させて構わないので、ICMP プロトコルのパケットは全て通過させます (ICMP にはポートの概念はありません)。

プロトコル	送信元アドレス	宛先アドレス	送信元ポート	宛先ポート
Any	Any	127.0.0.1	Any	Any
ICMP	Any	10.20.142.6	—	—

この節では、前述のルールを iptables コマンドで設定します。新しいルールの追加は以下のような書式で iptables コマンドを実行します。

```
iptables -A チェイン名 ルール
```

次にルールに適用するパケットの条件を指定します。条件は以下のオプションを用いて指定します。

オプション	解説
-------	----

-p プロトコル	パケットのプロトコル(ICMP、TCP、UDP)を指定する
-d アドレス	宛先(destination)アドレスを指定する
-s アドレス	送信元(source)アドレスを指定する
--dport ポート	宛先ポート番号を指定する
--sport ポート	送信元ポート番号を指定する
-j ターゲット	ターゲットを指定する

例えば、「送信元アドレスが www.example.net で、送信先ポート番号が 80 番の TCP パケットを通過させる」ためには、以下のようなコマンドを実行します。

```
# iptables -A INPUT -p tcp -s www.example.net --dport 80 -j ACCEPT
```

【練習】

1. 隣の方からのアクセスを全て拒否するように指定します。

```
iptables -A INPUT -s 隣の方のアドレス -j DROP
```

2. 「iptables -L」により、1.の状態を確認します。この例では 10.20.142.200 が隣のアドレスとしています。

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
DROP      all  --  10.20.142.200         anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

3. 隣の方からアクセスできないことを確認します。

ルールの適用を間違えた場合などは、該当するルールを削除するために「-D」オプションを用います。基本的には、ルールの作成の時に実行したコマンドの「-A」オプションを「-D」オプションに置き換えるとそのルールを削除できます。

```
# iptables -D INPUT -p tcp -s www.example.net --dport 80 -j ACCEPT
```

また、各チェーンの中でのテーブル上の行番号を指定することでそのルールを削除することもできます。例えば、INPUT チェインのポリシーの6番目のルールを削除したい場合は以下のように実行します。

```
# iptables -D INPUT 6
```

さらに、各チェーン内のそれぞれ全てのルールを削除する場合は「-F」オプションでまとめて削除できます。また、全チェーンを対象にする場合はチェーン名を省略します。

```
# iptables -F INPUT
# iptables -F
```

最後に、せっかく設定したルールも、保存せずにシステム停止すると、破棄されてしまいます。設定したルールを保存するためには、iptables-save コマンドにより保存します。

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

10.5 確認問題

これまでに確認してきたフィルタリングの方針に基づいて、以下の仕様を満たすようなパケットフィルタリングの設定を行います。

1. INPUT チェインの設定
フィルタリングポリシーは「DROP」、サービスに関するパケットのルールは下記を参照
2. FORWARD チェインの設定
フィルタリングポリシーは「DROP」
3. OUTPUT チェインの設定
フィルタリングポリシーは「ACCEPT」
4. 設定が完了したら、Apache、BIND、Postfixなどを起動し、自ホスト上のこれらのサービスにアクセスできるか？インターネット上の名前解決ができるか？隣の方のホストとの通信は可能か？などの確認を行います。

INPUTチェインのルール

#	プロトコル	送信元アドレス	宛先アドレス	送信元ポート	宛先ポート
1	Any	Any	ローカルホスト	Any	Any
2	ICMP	Any	自ホスト	—	—
3	TCP	Any	自ホスト	Any	80
4	TCP	Any	自ホスト	Any	53
5	UDP	Any	自ホスト	Any	53
6	TCP	Any	自ホスト	Any	25
7	TCP	Any	自ホスト	Any	110
8	TCP	Any	自ホスト	Any	22
9	TCP	Any	自ホスト	80	Any
10	TCP	Any	自ホスト	53	Any
11	UDP	Any	自ホスト	53	Any
12	TCP	Any	自ホスト	25	Any
13	TCP	Any	自ホスト	110	Any
14	TCP	Any	自ホスト	22	Any

付録 確認問題 解答例

付録 確認問題 解答例

第 1 章

1.1.3 練習

```

1.
[student@h006 ~]$ rpm -qa | grep "openss[h1]"
openssh-server-7.4p1-16.el7.x86_64
openssl-libs-1.0.2k-12.el7.x86_64
openssh-7.4p1-16.el7.x86_64
openssh-clients-7.4p1-16.el7.x86_64
openssl-1.0.2k-12.el7.x86_64
2.
[student@h006 ~]$ rpm -qi openssh
Name       : openssh
Version    : 7.4p1
Release    : 16.el7
Architecture: x86_64
(省略)

```

1.2.3 練習

講師マシン 10.20.121.6、自マシン 10.20.121.10 として

```

1.
[student@h010 ~]$ ip -4 addr show enp0s25      先に自 ID 確認
2: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
1000
    inet 10.20.121.10/16 brd 10.20.255.255 scope global enp0s25
        valid_lft forever preferred_lft forever
[student@h010 ~]$ ssh 10.20.121.6
The authenticity of host '10.20.121.6 (10.20.121.6)' can't be established.
ECDSA key fingerprint is SHA256:/LD2qACS/h0MWAcuZ+oNnQMNdjFdYJ3eB6jizWengUA.
ECDSA key fingerprint is MD5:ae:78:d0:f5:eb:bd:ee:42:54:20:a6:e4:77:e7:93:76.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.20.121.6' (ECDSA) to the list of known hosts.
student@10.20.121.6's password:
Last login: Thu Aug 16 11:13:58 2018
2.
[student@h006 ~]$ ip -4 addr show enp0s25
2: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    inet 10.20.121.6/16 brd 10.20.255.255 scope global noprefixroute enp0s25
        valid_lft forever preferred_lft forever
3.
[student@h006 ~]$ w
14:29:33 up 3:16, 3 users, load average: 0.06, 0.08, 0.19
USER      TTY      FROM          LOGIN@      IDLE        JCPU   PCPU   WHAT

```

```

student :0 :0 11:14 ?xdm? 15:00 0.27s /usr/libexec/gnome-sessio
student pts/0 :0 11:14 4:37 0.22s 0.22s bash
student pts/1 h010.s121.la.net 14:295.00s 0.04s 0.00s w
4.
[student@h006 ~]$ exit
ログアウト
Connection to 10.20.121.6 closed.
5.
[student@h010 ~]$ cat ~/.ssh/known_hosts
10.20.121.6 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBcTfSuIctwV6blMk8kvL07MRbyiie02Hj
A0bZqeR0c/Ri4SEikIO5Bifpq9j0GQt0diqx0rudIFe6zGB3zx8XE=

```

1.3.1 練習

```

1.
[root@h006 ~]# ps aux | grep sshd
root 1743 0.0 0.4 112812 4284 ? Ss 10:36 0:00 /usr/sbin/sshd -D
root 1822 0.0 0.6 161420 6104 ? Ss 10:37 0:00 sshd: student [pr
student 1824 0.0 0.2 161420 2344 ? S 10:37 0:00 sshd: student@pts
root 1924 0.0 0.0 112704 940 pts/1 R+ 10:47 0:00 grep sshd
2.
[root@h006 ~]# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
PID/Program name
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 1743/sshd
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 1056/master
tcp 0 0 192.168.56.101:22 192.168.56.1:50946 ESTABLISHED 1822/sshd: stude
tcp6 0 0 :::22 :::* LISTEN 1743/sshd
tcp6 0 0 :::1:25 :::* LISTEN 1056/master
[root@h006 ~]# ss -antp
State Recv-Q Send-Q Local Address:Port Peer Address:Port
LISTEN 0 128 *:22 *:22 users:
(("sshd",pid=1743,fd=3))
LISTEN 0 100 127.0.0.1:25 *:22 users:
(("master",pid=1056,fd=13))
ESTAB 0 0 192.168.56.101:22 192.168.56.1:50946 users:
(("sshd",pid=1824,fd=3),("sshd",pid=1822,fd=3))
LISTEN 0 128 :::22 :::* users:
(("sshd",pid=1743,fd=4))
LISTEN 0 100 :::1:25 :::* users:
(("master",pid=1056,fd=14))
3.
[root@h006 ~]# systemctl stop sshd
4.
[root@h006 ~]# ps aux | grep sshd

```

```

root 1937 0.0 0.0 112704 936 pts/1 R+ 10:48 0:00 grep sshd
[root@h006 ~]# ss -antp
State      Recv-Q Send-Q Local Address:Port      Peer Address:Port
LISTEN    0      100  127.0.0.1:25          *:*                    users:
(("master",pid=1056,fd=13))
LISTEN    0      100  :::1:25              :::*                    users:
(("master",pid=1056,fd=14))
5.
[root@h006 ~]# systemctl start sshd
6.
[root@h006 ~]# ps aux | grep sshd
root 1945 0.0 0.4 112812 4288 ? Ss 10:48 0:00 /usr/sbin/sshd
root 1947 0.0 0.0 112704 936 pts/1 S+ 10:48 0:00 grep sshd
[root@h006 ~]# ss -antp
State      Recv-Q Send-Q Local Address:Port      Peer Address:Port
LISTEN    0      128  *:22                  *:*                    users:
(("sshd",pid=1945,fd=3))
LISTEN    0      100  127.0.0.1:25          *:*                    users:
(("master",pid=1056,fd=13))
LISTEN    0      128  :::22                 :::*                    users:
(("sshd",pid=1945,fd=4))
LISTEN    0      100  :::1:25              :::*                    users:
(("master",pid=1056,fd=14))
7.
[root@h006 ~]# su - student
Last login: Fri Aug 17 10:37:56 JST 2018 from 192.168.56.1 on pts/0
[student@h006 ~]$ ssh localhost
The authenticity of host 'localhost (:::1)' can't be established.
ECDSA key fingerprint is SHA256:zNq6dmE5Msnx8g/i8AJ5gHRg5DgRmNwwRJtuUJ2zXkA.
ECDSA key fingerprint is MD5:37:db:af:dc:6b:e6:4c:ca:b9:fb:b7:65:79:96:d3:0d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
student@localhost's password:
Last failed login: Fri Aug 17 10:49:18 JST 2018 from localhost on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Fri Aug 17 10:48:58 2018 from 192.168.56.1
8.
[student@h006 ~]$ exit
□グアウト
Connection to localhost closed.
9.
[student@h006 ~]$ cat .ssh/known_hosts
localhost ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTI6bnVpdHhAYnTYAAAIbmlzdHhAYnTYAAABBBN01It6svkkfvrhBzUc9666+aDbhJ8eVo9
mju0nYs+W4k6ShH/H03TX0C1HVFy7w2KEXS1UXLKCRTEtit0bwP8U=

```

1.3.2 練習

```
1.
[root@h006 ~]# cat /etc/ssh/sshd_config
(省略)

2.
[root@h006 ~]# grep PermitRoot /etc/ssh/sshd_config
#PermitRootLogin yes
# the setting of "PermitRootLogin without-password".

3.
[root@h006 student]# ssh root@localhost
The authenticity of host 'localhost (:::1)' can't be established.
ECDSA key fingerprint is SHA256:zNq6dmE5Msnx8g/i8AJ5gHRg5DgRmNwwRJtuUJ2zXkA.
ECDSA key fingerprint is MD5:37:db:af:dc:6b:e6:4c:ca:b9:fb:b7:65:79:96:d3:0d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
root@localhost's password:
Last login: Fri Aug 17 11:06:32 2018
[root@h006 ~]# exit
ログアウト
Connection to localhost closed.

4.
[root@h006 ~]# vi /etc/ssh/sshd_config
(指定箇所を修正)
[root@h006 ~]# grep PermitRoot /etc/ssh/sshd_config
#PermitRootLogin yes
PermitRootLogin no      ← 追加内容
# the setting of "PermitRootLogin without-password".

5.
[root@h006 ~]# systemctl restart sshd

6.
[root@h006 ~]# ssh localhost
root@localhost's password:      ← 正しく入力しても、3回失敗する
Permission denied, please try again.
root@localhost's password:
Permission denied, please try again.
root@localhost's password:
Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
```

1.3.3 練習

```
1.
自ホスト 10.20.121.6、127.0.0.1、隣マシン 10.20.121.10 として 127.0.0.1 のみ許可する。

2.
[root@h006 ~]# cat /etc/hosts.allow
sshd: 127.0.0.1
[root@h006 ~]# cat /etc/hosts.deny
```

```
sshd: ALL
3.
[student@h010 ~]$ ssh 10.20.121.6
ssh_exchange_identification: read: Connection reset by peer
4.
[root@h006 ~]# cat /etc/hosts.allow
# sshd: 127.0.0.1
[root@h006 ~]# cat /etc/hosts.deny
# sshd: ALL
5.
[student@h010 ~]$ ssh 10.20.121.6
student@10.20.121.6's password:
Last login: Thu Aug 23 11:45:26 2018 from h010.s121.la.net
[student@h006 ~]$ exit
ログアウト
Connection to 10.20.121.6 closed.
[student@h010 ~]$
```

1.4 確認問題

1. SSH が推奨される理由

従来使用されていた TELNET はデータをそのまま(平文)で流してしまい、情報を盗聴される危険性がある。特にリモートログインするためのユーザー名、パスワードといった重要な情報が盗聴されるとアカウントを乗っ取られる可能性がある。

2. サーバー「なりすまし」を防ぐ工夫

ホスト間認証を行なっている。具体的には、初回の開始時にサーバーから公開鍵を送付クライアントで保持。攻撃者がなりすましたサーバーに接続すると、先に保持した公開鍵が一致せず、「なりすまし」を検出できる。

以下、3以降のサンプル

```
3.
[root@h006 ~]# yum update
読み込んだプラグイン:fastestmirror
Loading mirror speeds from cached hostfile
 * base: ftp.riken.jp
 * centosplus: ftp.riken.jp
 * epel: ftp.jaist.ac.jp
 * extras: ftp.riken.jp
 * updates: ftp.riken.jp
No packages marked for update
4.
[root@h006 ~]# vi /etc/ssh/sshd_config
(修正箇所のみ)
PermitRootLogin no
PasswordAuthentication yes
```

```
[root@h006 ~]# vi /etc/hosts.allow
sshd: 10.20.0.0/16, 127.0.0.1
[root@h006 ~]# vi /etc/hosts.deny
sshd: ALL
5.
[root@h006 ~]# systemctl restart sshd
6.
[root@h006 ~]# useradd sshuser
[root@h006 ~]# passwd sshuser
ユーザー sshuser のパスワードを変更。
新しいパスワード:
よくないパスワード: このパスワードは 8 未満の文字列です。
新しいパスワードを再入力してください:
passwd: すべての認証トークンが正しく更新できました。
7.
[student@h006 ~]$ ssh sshuser@10.20.121.6
sshuser@10.20.121.6's password:
Last failed login: Tue Aug 21 14:38:19 JST 2018 from h006.s121.la.net on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Tue Aug 21 14:38:02 2018 from h006.s121.la.net
[sshuser@h006 ~]$ exit
ログアウト
Connection to 10.20.121.6 closed.
[student@h006 ~]$ ssh sshuser@127.0.0.1
ssh_exchange_identification: read: Connection reset by peer
[student@h006 ~]$ ssh root@10.20.121.6
root@10.20.121.6's password:
Permission denied, please try again.
root@10.20.121.6's password:
Permission denied, please try again.
root@10.20.121.6's password:
Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
8.
[student@h010 ~]$ ssh sshuser@10.20.121.6
sshuser@10.20.121.6's password:
Last failed login: Tue Aug 21 14:40:29 JST 2018 from h010.s121.la.net on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Tue Aug 21 14:38:24 2018 from h006.s121.la.net
[sshuser@h006 ~]$ exit
ログアウト
Connection to 10.20.121.6 closed.
[student@h010 ~]$ ssh root@10.20.121.6
root@10.20.121.6's password:
Permission denied, please try again.
root@10.20.121.6's password:
Permission denied, please try again.
```

```
root@10.20.121.6's password:
```

```
Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
```

第 2 章

2.1.3 練習

1. 秘密鍵と公開鍵の2種類
2. 公開鍵。SSHに限らず、手元から離す鍵は公開鍵のみ

2.2.3 練習

```

1.
[student@h006 ~]$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/student/.ssh/id_dsa.
Your public key has been saved in /home/student/.ssh/id_dsa.pub.
The key fingerprint is:
SHA256:XIgnCf+536bxai40cKQzs9QHLHIPQD3xe4EGpG8up+o student@h006.s143.la.net
The key's randomart image is:
+----[DSA 1024]-----+
| .+++o                |
|                      |
| (省略)                |
| .E..  +=+o          |
+----[SHA256]-----+
2.
[student@h006 ~]$ cd .ssh
[student@h006 .ssh]$ cat id_dsa.pub >> authorized_keys
3.
[student@h006 .ssh]$ chmod 600 authorized_keys
[student@h006 .ssh]$ ls -l authorized_keys
-rw----- 1 student student 614  8月 18 13:32 authorized_keys
4.
[student@h006 ~]$ ssh localhost
Enter passphrase for key '/home/student/.ssh/id_dsa': ←パスワード入力
Last login: Sat Aug 18 14:09:25 2018 from 192.168.56.1

```

2.3.1 練習

自 IP が 10.20.121.6、隣が 10.20.121.10 の場合

```

[student@h006 ~]$ scp student@10.20.121.10:.bashrc ~/bash.cp
student@10.20.121.10's password:
.bashrc                                100% 280    12.2KB/s   00:00
[[student@h006 ~]$ ls -l bash.cp
-rw-r--r-- 1 student student 280  8月 18 14:18 bash.cp

```

2.3.2 練習

```

1.
[root@h006 student]# grep sftp /etc/ssh/etc/ssh/sshd_config
/etc/ssh/sshd_config:Subsystem sftp /usr/libexec/openssh/sftp-server
2.
再起動は不要
3.
[root@h006 student]# sftp student@localhost
student@localhost's password:
Connected to localhost.
4.
sftp> put /etc/hosts myhosts
Uploading /etc/hosts to /home/student/myhosts
/etc/hosts                               100% 185    564.8KB/s   00:00
sftp> get .bashrc bash.student
Fetching /home/student/.bashrc to bash.student
/home/student/.bashrc                     100% 280    574.8KB/s   00:00
sftp> quit

```

2.4 確認問題

1. 公開鍵暗号認証のメリット

アカウント情報をネットワークに流さずに確実なユーザー認証を行うことができるため、第三者によって送信される情報を解読される可能性がなくなる。

2. 公開鍵を用いた認証を行う操作手順

- 1) クライアントでキーペア(秘密鍵と公開鍵)を作成する。
- 2) ログイン先サーバーに、自分の公開鍵を登録する。
- 3) ログイン先サーバで、(2)のファイルを ~/.ssh/authorized_keys へ追記
- 4) (3)のパーミッションを調整する。

```

~/.ssh          700
authorized_keys 600

```

3 以降の通り

```

3.
[root@h010 ~]# useradd sshguest
[root@h010 ~]# passwd sshguest
ユーザー sshguest のパスワードを変更。
新しいパスワード:
よくないパスワード: このパスワードは 8 未満の文字列です。
新しいパスワードを再入力してください:
passwd: すべての認証トークンが正しく更新できました。
4.
[student@h006 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
(省略)
5.

```

```
[student@h006 ~]$ scp .ssh/id_rsa.pub sshguest@10.20.121.10:id.student
The authenticity of host 'localhost (:::1)' can't be established.
ECDSA key fingerprint is SHA256:/LD2qACS/h0MwAcUz+oNnQMNdjFdYJ3eB6jizWengUA.
ECDSA key fingerprint is MD5:ae:78:d0:f5:eb:bd:ee:42:54:20:a6:e4:77:e7:93:76.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
sshguest@10.20.121.10's password:
id_rsa.pub                                100% 406   269.8KB/s   00:00
6.
[student@h006 ~]$ ssh sshguest@10.20.121.10
sshguest@10.20.121.10's password:
7.
[sshguest@h010 ~]$ mkdir .ssh
[sshguest@h010 ~]$ cat id.student >> .ssh/authorized_keys
8.
[sshguest@h010 ~]$ chmod 700 .ssh
[sshguest@h010 ~]$ chmod 600 .ssh/authorized_keys
9.
[sshguest@h010 ~]$ exit
ログアウト
Connection to 10.20.121.10 closed.
[student@h006 ~]$
10.
[student@h006 ~]$ ssh sshguest@10.20.121.10
Enter passphrase for key '/home/student/.ssh/id_rsa':
Last login: Wed Aug 22 10:49:56 2018 from h006.s121.la.net
[sshguest@h010 ~]$
```

第 3 章

3.1.3 練習

検索エンジンで「RFC5034」「RFC1731」を検索し内容を確認する。

3.2.3 練習

```
[sshguest@h006 ~]$ nslookup -query=mx i.softbank.jp
Server:          10.20.250.1
Address:         10.20.250.1#53

Non-authoritative answer:
i.softbank.jp   mail exchanger = 10 msv.softbank.jp.

Authoritative answers can be found from:
softbank.jp     nameserver = dns06.bbtec.net.
softbank.jp     nameserver = dns04.bbtec.net.
softbank.jp     nameserver = dns03.bbtec.net.
msv.softbank.jp internet address = 117.46.9.104
msv.softbank.jp internet address = 117.46.11.72
msv.softbank.jp internet address = 117.46.5.72
msv.softbank.jp internet address = 117.46.7.40
dns06.bbtec.net internet address = 219.188.193.164
dns04.bbtec.net internet address = 219.188.198.28
dns03.bbtec.net internet address = 221.110.34.4
```

3.3.2 練習

メール本文の From (差出人)、To (宛先) は詐称可能なので、「誰が誰に宛てたメールか」を証明することはできない。

3.4.2 練習

POP 3 は暗号化されておらず、ユーザ名、パスワードがそのままインターネット上を流れるため情報漏洩の危険がある。よって IMAP を利用すべき。

第 4 章

4.1.2 練習

```
[root@h006 ~]# less /etc/postfix/main.cf
```

(関連箇所のみ抜粋)

```
[root@h006 ~]# postconf -n
inet_interfaces = all
mydestination = $myhostname
mydomain = s121.la.net
myhostname = h006.s121.la.net
mynetworks = 10.20.0.0/16, 127.0.0.0/8
myorigin = $myhostname
```

4.2.1 練習

```
1.
[root@h006 ~]# systemctl start postfix
2.
[root@h006 ~]# ps aux |grep postfix
root      2143  0.0  0.2  94300  2088 ?        Ss   15:38   0:00
/usr/libexec/postfix/master -w
postfix   2144  0.0  0.4  94412  4092 ?        S    15:38   0:00 pickup -l -t unix -u
postfix   2145  0.0  0.4  94476  4124 ?        S    15:38   0:00 qmgr -l -t unix -u
root      2155  0.0  0.0  112720  948 pts/0    R+   15:42   0:00 grep postfix
3.
[root@h006 ~]# pstree -p
systemd(1)─NetworkManager(623)─dhclient(667)
      (省略)
          └─master(2143)─pickup(2144)
              |
              └─qmgr(2145)
      (省略)
```

4.2.2 練習

```
1.
[student@h006 ~]$ mail student@h006.s121.la.net
Subject: test mail 4.2.1
Test message
.
EOT
2.
[student@h006 ~]$ mail
Heirloom Mail version 12.5 7/5/10. Type ? for help.
"/var/spool/mail/student": 1 message 1 new
```

```
>N 1 student                      Wed Aug 22 13:04 18/628  "test mail 4.2.1"
& 1
Message 1:
From student@h006.s121.la.net  Wed Aug 22 13:04:06 2018
Return-Path: <student@h006.s121.la.net>
X-Original-To: student@h006.s121.la.net
Delivered-To: student@h006.s121.la.net
Date: Wed, 22 Aug 2018 13:04:06 +0900
To: student@h006.s121.la.net
Subject: test mail 4.2.1
User-Agent: Heirloom mailx 12.5 7/5/10
Content-Type: text/plain; charset=us-ascii
From: student@h006.s121.la.net (student)
Status: R

Test message

& q
Held 1 message in /var/spool/mail/student
```

```
3.
[root@h006 ~]# telnet localhost 25
Trying ::1...
Connected to localhost.
Escape character is '^]'.
220 h006.s121.la.net ESMTP Postfix
ehlo test
250-h006.s121.la.net
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
mail from: yakoshi@h006.s121.la.net
250 2.1.0 Ok
rcpt to: root@localhost
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: Telnet mail
test mail
.
250 2.0.0 Ok: queued as A5030DBD11
```

```
quit
221 2.0.0 Bye
Connection closed by foreign host.
メールが /var/spool/mail/root にあります
[root@h006 ~]# tail /var/spool/mail/root
Received: from test (localhost [IPv6:::1])
    by h006.s121.la.net (Postfix) with ESMTP id A5030DBD11
    for <root@localhost>; Sat, 18 Aug 2018 15:50:36 +0900 (JST)
Subject: Telnet mail
Message-Id: <20180818065047.A5030DBD11@h006.s121.la.net>
Date: Sat, 18 Aug 2018 15:50:36 +0900 (JST)
From: yakoshi@h006.s121.la.net

test mail
```

```
4-a)
[student@h006 ~]$ telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 h006.s121.la.net SMTP Postfix
ehlo test
250-h006.s121.la.net
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
mail from:student
250 2.1.0 Ok
rcpt to:student@h010.s121.la.net
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: 4.2.2 4-a

Test
.
250 2.0.0 Ok: queued as B26E360F4C7A
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

```
(隣のマシンにて)
新しいメールが /var/spool/mail/student にあります
[student@h010 ~]$ mail
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/spool/mail/student": 1 message 1 new
>N 1 student@h006.s121.la Wed Aug 22 11:43 17/703 "4.2.2 4-a"
& 1
Message 1:
From student@h006.s121.la.net Wed Aug 22 11:43:52 2018
Return-Path: <student@h006.s121.la.net>
X-Original-To: student@h010.s121.la.net
Delivered-To: student@h010.s121.la.net
Subject: 4.2.2 4-a
Date: Wed, 22 Aug 2018 11:41:19 +0900 (JST)
From: student@h006.s121.la.net
Status: R

Test

& q
Held 1 message in /var/spool/mail/student
```

```
4-b)
[student@h006 ~]$ telnet 10.20.121.10 25
Trying 10.20.121.10...
Connected to 10.20.121.10.
Escape character is '^]'.
220 h010.s121.la.net ESMTP Postfix
ehlo test
250-h010.s121.la.net
    (省略)
250 DSN
mail from:student@h006.s121.la.net
250 2.1.0 Ok
rcpt to:student@h010.s121.la.net
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: 4.2.2 4-b

Test 2
.
250 2.0.0 Ok: queued as 36A3A600009F
quit
```

```
221 2.0.0 Bye
Connection closed by foreign host.
    (隣マシンにて)
新しいメールが /var/spool/mail/student にあります
[student@h010 ~]$ mail
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/spool/mail/student": 2 messages 1 new
    1 student@h006.s121.la  Wed Aug 22 11:43  18/714  "4.2.2 4-a"
>N  2 student@h006.s121.la  Wed Aug 22 11:45  11/385  "4.2.2 4-b"
& 2
Message  2:
From student@h006.s121.la.net  Wed Aug 22 11:45:56 2018
Return-Path: <student@h006.s121.la.net>
X-Original-To: student@h010.s121.la.net
Delivered-To: student@h010.s121.la.net
Subject: 4.2.2 4-b
Status: R

Test 2

& q
Held 2 messages in /var/spool/mail/student
```

```
4-c)
[student@h006 ~]$ telnet 10.20.121.10 25
Trying 10.20.121.10...
Connected to 10.20.121.10.
Escape character is '^]'.
220 h010.s121.la.net ESMTP Postfix
ehlo test
250-h010.s121.la.net
    (省略)
250 DSN
mail from:student
250 2.1.0 Ok
rcpt to:student@h006.s121.la.net
454 4.7.1 <student@h006.s121.la.net>: Relay access denied
    リレーが拒絶されメール送信できない。
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

4.2.3 練習

```
[root@h006 ~]# tail /var/log/maillog
Aug 18 15:50:16 h006 postfix/smtpd[2213]: connect from localhost[::1]
Aug 18 15:50:47 h006 postfix/smtpd[2213]: A5030DBD11: client=localhost[::1]
Aug 18 15:51:06 h006 postfix/cleanup[2217]: A5030DBD11: message-
id=<20180818065047.A5030DBD11@h006.s121.la.net>
Aug 18 15:51:06 h006 postfix/qmgr[2145]: A5030DBD11: from=<yakoshi@h006.s121.la.net>,
size=333, nrcpt=1 (queue active)
Aug 18 15:51:06 h006 postfix/local[2218]: A5030DBD11: to=<root@localhost.s121.la.net>,
orig_to=<root@localhost>, relay=local, delay=30, delays=30/0.01/0/0, dsn=2.0.0,
status=sent (delivered to mailbox)
Aug 18 15:51:06 h006 postfix/qmgr[2145]: A5030DBD11: removed
Aug 18 15:51:08 h006 postfix/smtpd[2213]: disconnect from localhost[::1]
Aug 18 15:54:28 h006 postfix/anvil[2215]: statistics: max connection rate 1/60s for
(smtp:::1) at Aug 18 15:50:16
Aug 18 15:54:28 h006 postfix/anvil[2215]: statistics: max connection count 1 for
(smtp:::1) at Aug 18 15:50:16
Aug 18 15:54:28 h006 postfix/anvil[2215]: statistics: max cache size 1 at Aug 18
15:50:16
```

4.3.1 練習

```
1.
[root@h006 ~]# date | mail -s "no connect" student@example.net
2.
[root@h006 ~]# mailq
-Queue ID- --Size-- ----Arrival Time---- -Sender/Recipient-----
9DFA9838E      467 Sat Aug 18 16:10:54 root@h006.s121.la.net
                (connect to example.net[93.184.216.34]:25: Connection refused)
                student@example.net

-- 0 Kbytes in 1 Request.
3.
[root@h006 ~]# find /var/spool/postfix/ -name 9DFA9838E -ls
4498878    4 -rw-----    1 postfix postfix          223  8月 18 16:10
/var/spool/postfix/defer/9/9DFA9838E
 33678    4 -rwx-----    1 postfix postfix          706  8月 18 16:15
/var/spool/postfix/deferred/9/9DFA9838E
4.
[root@h006 ~]# postsuper -d 9DFA9838E
postsuper: 9DFA9838E: removed
postsuper: Deleted: 1 message
5.
[root@h006 ~]# mailq
Mail queue is empty
```

4.4.1 練習

```
1.
```

```

[root@h006 ~]# vi /etc/aliases
[root@h006 ~]# tail /etc/aliases
(省略)
# Person who should get root's mail
#root:          marc
root:           student
2.
[root@h006 ~]# newaliases
3.
[root@h006 ~]# date | mail -s "alias test" root
4.
[root@h006 ~]# tail /var/spool/mail/student
Subject: alias test
User-Agent: Heirloom mailx 12.5 7/5/10
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit
Message-Id: <20180818071638.A8822DBD0F@h006.s121.la.net>
From: root@h006.s121.la.net (root)

2018年  8月 18日 土曜日 16:16:38 JST

```

4.4.3 練習

```

[student@h006 ~]$ vi ~/.forward
[student@h006 ~]$ cat ~/.forward
ycos001@yahoo.co.jp
\student
[student@h006 ~]$ date | mail -s "Forward test" student

```



第5章

5.2.1 練習

```
1.
[root@h006 ~]# systemctl status postfix
● postfix.service - Postfix Mail Transport Agent
   Loaded: loaded (/usr/lib/systemd/system/postfix.service; enabled; vendor preset:
disabled)
   Active: active (running) since 土 2018-08-18 15:38:49 JST; 52min ago
 Main PID: 2143 (master)
   CGroup: /system.slice/postfix.service
           └─2143 /usr/libexec/postfix/master -w
             └─2144 pickup -l -t unix -u
               └─2145 qmgr -l -t unix -u
      (省略)

2.
[root@h006 ~]# yum -y install dovecot
      (省略)
依存性関連をインストールしました:
      clucene-core.x86_64 0:2.3.3.4-11.el7

完了しました!

3.
[root@h006 ~]# rpm -ql dovecot
/etc/dovecot
/etc/dovecot/conf.d
/etc/dovecot/conf.d/10-auth.conf
/etc/dovecot/conf.d/10-director.conf
      (省略)
```

5.2.3 練習

```
1.
[root@h006 ~]# vi /etc/dovecot/dovecot.conf
      (修正箇所抜粋)
# Protocols we want to be serving.
#protocols = imap pop3 lmtp
protocols = pop3

2.
[root@h006 ~]# vi /etc/dovecot/conf.d/10-mail.conf
      (修正箇所抜粋)
# See doc/wiki/Variables.txt for full list. Some examples:
#
# mail_location = maildir:~/Maildir
# mail_location = mbox:~/mail:INBOX=/var/mail/%u
# mail_location = mbox:/var/mail/%d/%1n/%n:INDEX=/var/indexes/%d/%1n/%n
```

```
#
# <doc/wiki/MailLocation.txt>
#
#mail_location =
mail_location = mbox:~/mail:INBOX=/var/mail/%u

# Grant access to these supplementary groups for mail processes. Typically
# these are used to set up access to shared mailboxes. Note that it may be
# dangerous to set these if users can create symlinks (e.g. if "mail" group is
# set here, ln -s /var/mail ~/mail/var could allow a user to delete others'
# mailboxes, or ln -s /secret/shared/box ~/mail/mybox would allow reading it).
#mail_access_groups =
mail_access_groups = mail

3.
[root@h006 ~]# systemctl start dovecot

4-8.
[root@h006 ~]# telnet localhost 110
Trying ::1...
Connected to localhost.
Escape character is '^]'.
+OK Dovecot ready.

user student
+OK

pass himitu
+OK Logged in.

list
+OK 10 messages:
1 593
    (省略)
10 592
.

retr 10
+OK 592 octets
Return-Path: <student@h006.s121.la.net>
X-Original-To: student
Delivered-To: student@h006.s121.la.net
Received: by h006.s121.la.net (Postfix, from userid 1000)
    id BE7C2838E; Sat, 18 Aug 2018 16:20:47 +0900 (JST)
Date: Sat, 18 Aug 2018 16:20:47 +0900
To: student@h006.s121.la.net
Subject: forward test
User-Agent: Heirloom mailx 12.5 7/5/10
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit
Message-Id: <20180818072047.BE7C2838E@h006.s121.la.net>
```

```
From: student@h006.s121.la.net (LA Standard)
```

```
2018年 8月 18日 土曜日 16:20:47 JST
```

```
.
```

```
quit
```

```
+OK Logging out.
```

```
Connection closed by foreign host.
```

5.3.1 練習

```
[root@h006 ~]# cd /usr/local/src
[root@h006 src]# wget https://osdn.net/dl/sylpheed/sylpheed-3.7.0.tar.bz2
--2018-08-18 16:46:55-- https://osdn.net/dl/sylpheed/sylpheed-3.7.0.tar.bz2
osdn.net (osdn.net) を DNS に問いあわせています... 202.221.179.17
(省略)
`sylpheed-3.7.0.tar.bz2' に保存中

100%[=====>] 3,612,328 1.39MB/s 時間 2.5s

2018-08-18 16:47:00 (1.39 MB/s) - `sylpheed-3.7.0.tar.bz2' へ保存完了 [3612328/3612328]

[root@h006 src]# tar xjf sylpheed-3.7.0.tar.bz2
[root@h006 src]# cd sylpheed-3.7.0
[root@h006 sylpheed-3.7.0]# ./configure
checking build system type... x86_64-unknown-linux-gnu
(省略)
checking pkg-config is at least version 0.16... yes
checking for GLIB - version >= 2.8.0... no
*** Could not run GLIB test program, checking why...
*** The test program failed to compile or link. See the file config.log for the
*** exact error that occurred. This usually means GLIB is incorrectly installed.
configure: error: Test for GLib failed. See the 'INSTALL' for help.
↑ パッケージ不足のため、エラー終了

[root@h006 sylpheed-3.7.0]# yum install gcc glib2-devel gtk2-devel
(省略)
zlib-devel.x86_64 0:1.2.7-17.el7

完了しました!
[root@h006 sylpheed-3.7.0]# ./configure
(省略)
sylpheed 3.7.0

GnuPG      : no
JPilot     : no
LDAP       : no
OpenSSL    : no
```

```
iconv      : yes
compface   : no
IPv6       : yes
GtkSpell   : no
Oniguruma  : no
GThread    : yes

The binary will be installed in /usr/local/bin

Configure finished, type 'make' to build.
[root@h006 sylpheed-3.7.0]# make
make all-recursive
make[1]: ディレクトリ `/usr/local/src/sylpheed-3.7.0' に入ります
(省略)
make[2]: ディレクトリ `/usr/local/src/sylpheed-3.7.0' から出ます
make[1]: ディレクトリ `/usr/local/src/sylpheed-3.7.0' から出ます
[root@h006 sylpheed-3.7.0]# make install
Making install in ac
(省略)
/bin/mkdir -p '/usr/local/share/applications'
/bin/install -c -m 644 sylpheed.desktop '/usr/local/share/applications'
make[2]: ディレクトリ `/usr/local/src/sylpheed-3.7.0' から出ます
make[1]: ディレクトリ `/usr/local/src/sylpheed-3.7.0' から出ます
[root@h006 sylpheed-3.7.0]#
```

5.3.2 練習

割愛

第6章

6.2.4 練習

```

1.
[root@h006 ~]# yum install nfs-utils
(省略)
[root@h006 ~]# systemctl start nfs-server
[root@h006 ~]# systemctl status nfs-server
• nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled; vendor p
(省略)

2.
[root@h006 ~]# mkdir /work

3.
[root@h006 ~]# vi /etc/exports
[root@h006 ~]# cat /etc/exports
/work    10.20.0.0/255.255.0.0()

4.
[root@h006 ~]# exportfs -a
[root@h006 ~]# exportfs
/work    10.20.0.0/255.255.0.0
(隣マシン)

5.
[root@h010 ~]# mkdir /media/nfs

6.
[root@h010 ~]# mount 10.20.121.6:/work /media/nfs

7.
[root@h010 ~]# df /media/nfs
ファイルシステム 1K-ブロック 使用 使用可 使用% マウント位置
10.20.121.6:/work 6486016 1924352 4561664 30% /media/nfs

8.
[root@h010 ~]# umount /media/nfs

```

6.2.5 練習

```

1.
[root@h006 ~]# vi /etc/exports
[root@h006 ~]# cat /etc/exports
/tmp    10.20.0.0/255.255.0.0(rw)

2.
[root@h006 ~]# exportfs -r
[root@h006 ~]# exportfs -v
/tmp    10.20.0.0/255.255.0.0(rw, sync, wdelay, hide, no_subtree_check, sec=sys, secure,
root_squash, no_all_squash)

3.
[root@h006 ~]# df /media/nfs

```

```

ファイルシス          1K-ブロック   使用  使用可  使用%  マウント位置
/dev/mapper/centos_h006-root    6486016 1924332 4561684   30% /
[root@h006 ~]# mount 10.20.121.6:/tmp /media/nfs
[root@h006 ~]# df /media/nfs
ファイルシス  1K-ブロック   使用  使用可  使用%  マウント位置
10.20.121.6:/tmp    6486016 1924224 4561792   30% /media/nfs
4.
[root@h006 ~]# cd /media/nfs
[root@h006 nfs]# touch write.test
[root@h006 nfs]# ls -l write.test
-rw-r--r-- 1 nfsnobody nfsnobody 0  8月 18 17:38 write.test
[root@h010 nfs]# cd
[root@h010 ~]# umount /media/nfs

```

6.4 確認問題

```

1.
[root@h006 ~]# df
ファイルシス          1K-ブロック   使用  使用可  使用%  マウント位置
/dev/mapper/centos-root 46487372 4232804 42254568   10% /
devtmpfs                3917092      0 3917092    0% /dev
      (省略、NFS マウントがないことを確認)
2.
[root@h006 ~]# cat /etc/exports
/home  10.20.0.0/255.255.0.0(rw)
3.
[root@h006 ~]# exportfs -r
[root@h006 ~]# exportfs -v
/home
10.20.0.0/255.255.0.0(rw, sync, wdelay, hide, no_subtree_check, sec=sys, secure, root_squash,
no_all_squash)
4.
      (隣マシン)
[root@h010 ~]# mount 10.20.121.6:/home /media/nfs
[root@h010 ~]# ls -l /media/nfs
合計 4
drwx-----. 3 ito      ito      96  8月 22 10:47 sshguest
drwx-----. 5 abe      abe     128  8月 21 14:38 sshuser
drwx----x. 19 student student 4096  8月 22 13:12 student
[root@h010 ~]# id student
uid=1000(student) gid=1000(student) groups=1000(student)
[root@h010 ~]# id abe
uid=1001(abe) gid=1001(abe) groups=1001(abe)
      passwd 突合のち、UID の同じユーザーと違うユーザーを確認
      (自マシン)
[root@h006 ~]# ls -l /home
合計 4

```

```
drwx----- 3 sshguest sshguest 96  8月 22 10:47 sshguest
drwx----- 5 sshuser  sshuser 128  8月 21 14:38 sshuser
drwx-----x.19 student  student 4096  8月 22 13:12 student

[root@h006 ~]# id student
uid=1000(student) gid=1000(student) groups=1000(student)
[root@h006 ~]# id sshuser
uid=1001(sshuser) gid=1001(sshuser) groups=1001(sshuser)

6.
[root@h010 ~]# exit
ログアウト
[student@h010 ~]$ cd /media/nfs/student/
[student@h010 student]$ pwd
/media/nfs/student
[student@h010 student]$ ls -l
合計 1688
drwxr-xr-x. 2 student student      6  8月 14 10:22 Desktop
drwxr-xr-x. 2 student student      6  8月 14 10:22 Documents
drwxr-xr-x. 2 student student      6  8月 14 10:22 Downloads
(省略)

7.
[student@h010 student]$ touch h010.file

8.
[student@h010 student]$ ls -l h010.file
-rw-rw-r--. 1 student student 0  8月 22 14:05 h010.file

9.
[student@h010 student]$ cd ../sshuser
bash: cd: ../sshuser: 許可がありません

10
[student@h010 student]$ cd
[student@h010 ~]$ su -
パスワード:
[root@h010 ~]# umount /media/nfs
```

第7章

7.1.1 練習

```
[student@h006 ~]$ ftp ftp.riken.jp
Connected to ftp.riken.jp (134.160.38.1).
220 ::ffff:134.160.38.1 FTP server ready
Name (ftp.riken.jp:student): ftp
331 Anonymous login ok, send your complete email address as your password
Password:      ←自身のメールアドレスを入力（ノーチェック）
230-*****

ftp.riken.jp is an unsupported ftp/http/https/rsync
service of RIKEN Nishina Center for research support.
Use entirely at your own risk - no warranty
is expressed or implied.
Complaints and questions should be sent to
ftp-admin a.t. ftp.riken.jp
*****

230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd net/ProFTPD/distrib/source/
250 CWD command successful
ftp> ls
227 Entering Passive Mode (134,160,38,1,175,235)
150 Opening ASCII mode data connection for file list
(省略)
-rw-r--r--  1 archive  archive  20251898 Apr 10  2017 proftpd-1.3.6.tar.gz
-rw-r--r--  1 archive  archive      204 Apr 10  2017 proftpd-1.3.6.tar.gz.asc
-rw-r--r--  1 archive  archive      54 Apr 10  2017 proftpd-1.3.6.tar.gz.md5
226 Transfer complete
ftp> bin
200 Type set to I
ftp> prompt      ← mget のダウンロード確認をオフ
Interactive mode off.
ftp> mget proftpd-1.3.6.tar.*
local: proftpd-1.3.6.tar.gz.md5 remote: proftpd-1.3.6.tar.gz.md5
227 Entering Passive Mode (134,160,38,1,150,105)
150 Opening BINARY mode data connection for proftpd-1.3.6.tar.gz.md5 (54 bytes)
226 Transfer complete
54 bytes received in 0.0103 secs (5.26 Kbytes/sec)
(省略)
20251898 bytes received in 4.94 secs (4098.08 Kbytes/sec)
ftp> quit
221 Goodbye.
[student@h006 ~]$ ls pro*
```

```

proftpd-1.3.6.tar.gz  proftpd-1.3.6.tar.gz.ascproftpd-1.3.6.tar.gz.md5
[student@h006 ~]$ md5sum -c proftpd-1.3.6.tar.gz.md5
proftpd-1.3.6.tar.gz: 完了

```

7.1.2 練習

```

[root@h006 ~]# cd /usr/local/src
[root@h006 src]# tar xzf ~student/proftpd-1.3.6.tar.gz
[root@h006 src]# ls
proftpd-1.3.6
[root@h006 src]# cd proftpd-1.3.6/
[root@h006 proftpd-1.3.6]# ls
COPYING          README.PAM          acconfig.h        locale
CREDITS          README.Solaris2.5x  aclocal.m4        ltmain.sh
ChangeLogREADME.Unixware  config.guess       m4
INSTALL          README.capabilities config.h.in        modules
(省略)

```

7.1.3 練習

```

[root@h006 proftpd-1.3.6]# ./configure --prefix=/usr/local/proftpd
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
(省略)
config.status: executing default commands

-----
Build Summary
-----

Building the following static modules:
  mod_ident
  mod_cap

-----
[root@h006 proftpd-1.3.6]#

```

7.1.4 練習

```

[root@h006 proftpd-1.3.6]# make
echo \#define BUILD_STAMP "`date +%a %b %e %Y %H:%M:%S %Z`" > include/buildstamp.h
cd lib/ && make lib
make[1]: ディレクトリ `/usr/local/src/proftpd-1.3.6/lib' に入ります
gcc -DHAVE_CONFIG_H -DLINUX -I.. -I../include -g2 -O2 -Wall -fno-omit-frame-pointer
-c pr_fnmatch.c
(省略)
gcc -L./lib -rdynamic -o ftpwho utils/ftpwho.o utils/scoreboard.o utils/misc.o
-lsupp -lcrypt -ldl

```

```
[root@h006 proftpd-1.3.6]# make install
cd lib/ && make lib
make[1]: ディレクトリ `/usr/local/src/proftpd-1.3.6/lib' に入ります
(省略)
/bin/install -c -o root -g root -m 0644 proftpd.pc
/usr/local/proftpd/lib/pkgconfig/proftpd.pc
test -z "" || (cd locale/ && make install)

[root@h006 proftpd-1.3.6]# ls -l /usr/local/proftpd/
合計 0
drwxr-xr-x 2 root root 124  8月 19 07:50 bin
drwxr-xr-x 2 root root  26  8月 19 07:50 etc
drwxr-xr-x 3 root root  21  8月 19 07:50 include
drwxr-xr-x 4 root root  38  8月 19 07:50 lib
drwxr-xr-x 2 root root   6  8月 19 07:50 libexec
drwxr-xr-x 2 root root  70  8月 19 07:50 sbin
drwxr-xr-x 4 root root  31  8月 19 07:50 share
drwxr-xr-x 2 root root   6  8月 19 07:50 var
```

7.2.1 練習

```
[root@h006 proftpd-1.3.6]# vi /usr/local/proftpd/etc/proftpd.conf
[root@h006 proftpd-1.3.6]# grep nobody /usr/local/proftpd/etc/proftpd.conf
# "nobody" and "ftp" for normal operation and anon.
User                nobody
Group                nobody
```

7.2.2 練習

```
[root@h006 proftpd-1.3.6]# /usr/local/proftpd/sbin/proftpd
[root@h006 proftpd-1.3.6]# ps aux | grep proftpd
nobody  11508  0.0  0.1  44836  1408 ?        Ss   08:03   0:00 proftpd: (accepting
connections)
root    11535  0.0  0.0  112720  948 pts/0    R+   08:03   0:00 grep proftpd
[root@h006 proftpd-1.3.6]# ss -antp
State      Recv-Q  Send-Q  Local Address:Port      Peer Address:Port
(抜粋)
LISTEN    0       128          *:21                   *:*                   users:
(("proftpd",pid=11508,fd=0))

[root@h006 proftpd-1.3.6]# lsof -i:21
COMMAND  PID  USER  FD  TYPE  DEVICE  SIZE/OFF  NODE  NAME
proftpd  11508  nobody  0u  IPv4  22258      0t0  TCP  *:ftp (LISTEN)
[root@h006 proftpd-1.3.6]# kill 11508
[root@h006 proftpd-1.3.6]# lsof -i:21
[root@h006 proftpd-1.3.6]# ss -antp | grep :21
```

7.2.3 練習

```
1.
[root@h006 proftpd-1.3.6]# vi /usr/local/proftpd/etc/proftpd.conf
# Bar use of SITE CHMOD by default
<Limit SITE_CHMOD>
    DenyAll
</Limit>
<Limit Write>    ←3行追加
    DenyAll
</Limit>
[root@h006 proftpd-1.3.6]# kill 11595
[root@h006 proftpd-1.3.6]# /usr/local/proftpd/sbin/proftpd

2.
[root@h006 proftpd-1.3.6]# ftp 127.0.0.1
Connected to 127.0.0.1 (127.0.0.1).
220 ProFTPD Server (ProFTPD Default Installation) [127.0.0.1]
Name (127.0.0.1:root): student
331 Password required for student
Password:
230 User student logged in
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put /etc/hosts myhost
local: /etc/hosts remote: myhost
227 Entering Passive Mode (127,0,0,1,159,203).
550 myhost: operation not permitted    ←アップロードが拒否されている
ftp> quit
221 Goodbye.

3.
[root@h006 proftpd-1.3.6]# vi /usr/local/proftpd/etc/proftpd.conf
# ycos add
<Limit Write>
    Allow 10.20.0.0/16
    DenyAll
</Limit>

4.
[root@h006 proftpd-1.3.6]# kill 11604
[root@h006 proftpd-1.3.6]# /usr/local/proftpd/sbin/proftpd

5.
[root@h006 proftpd-1.3.6]# ftp 10.20.121.6
Connected to 10.20.121.6 (10.20.121.6).
220 ProFTPD Server (ProFTPD Default Installation) [10.20.121.6]
Name (10.20.121.6:root): student
331 Password required for student
Password:
230 User student logged in
Remote system type is UNIX.
```

```
Using binary mode to transfer files.
ftp> put /etc/hosts myhost ← 10.20.からはアップロードできる
local: /etc/hosts remote: myhost
227 Entering Passive Mode (10,20,121,6,158,76).
150 Opening BINARY mode data connection for myhost
226 Transfer complete
185 bytes sent in 0.000212 secs (872.64 Kbytes/sec)
ftp> quit
221 Goodbye.

[root@h006 proftpd-1.3.6]# ftp 127.0.0.1
Connected to 127.0.0.1 (127.0.0.1).
220 ProFTPD Server (ProFTPD Default Installation) [127.0.0.1]
Name (127.0.0.1:root): student
331 Password required for student
Password:
230 User student logged in
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put /etc/hosts myhost2
local: /etc/hosts remote: myhost2
227 Entering Passive Mode (127,0,0,1,160,93).
550 myhost2: Operation not permitted ← 127.0.0.1 からはアップロードが拒否される
6.
ftp> cd /
250 CWD command successful
ftp> ls
227 Entering Passive Mode (127,0,0,1,129,156).
150 Opening ASCII mode data connection for file list
lrwxrwxrwx  1 root    root          7 Jul 23 00:16 bin -> usr/bin
dr-xr-xr-x  5 root    root        4096 Aug 17 02:31 boot
      (省略)
ftp> quit
221 Goodbye.
```

7.2.4 練習

```
1.
[root@h006 proftpd-1.3.6]# vi /usr/local/proftpd/etc/proftpd.conf
[root@h006 proftpd-1.3.6]# grep Root /usr/local/proftpd/etc/proftpd.conf
#DefaultRoot ~
DefaultRoot ~/public_html
2.
[root@h006 proftpd-1.3.6]# lsof -i:21
COMMAND  PID  USER  FD  TYPE DEVICE SIZE/OFF  NODE NAME
proftpd 11642 nobody  0u  IPv4  24088      0t0  TCP *:ftp (LISTEN)
[root@h006 proftpd-1.3.6]# kill 11642
```

```
[root@h006 proftpd-1.3.6]# /usr/local/proftpd/sbin/proftpd
3.
[root@h006 proftpd-1.3.6]# exit
ログアウト
[student@h006 ~]$ mkdir public_html
[student@h006 ~]$ date > public_html/date.txt
4.
[student@h006 ~]$ ftp 127.0.0.1
Connected to 127.0.0.1 (127.0.0.1).
220 ProFTPD Server (ProFTPD Default Installation) [127.0.0.1]
Name (127.0.0.1:student):
331 Password required for student
Password:
230 User student logged in
Remote system type is UNIX.
Using binary mode to transfer files.
5.
ftp> ls
227 Entering Passive Mode (127,0,0,1,128,176).
150 Opening ASCII mode data connection for file list
-rw-rw-r--  1 student  student      43 Aug 18 23:36 date.txt
226 Transfer complete
ftp> cd /
250 CWD command successful
ftp> pwd
257 "/" is the current directory
ftp> ls
227 Entering Passive Mode (127,0,0,1,145,249).
150 Opening ASCII mode data connection for file list
-rw-rw-r--  1 student  student      43 Aug 18 23:36 date.txt
226 Transfer complete
ftp> quit
221 Goodbye.
```

7.2.5 練習

```
[root@h006 student]# vi /etc/ftpusers
[root@h006 student]# cat /etc/ftpusers
student
[root@h006 student]# ftp 127.0.0.1
Connected to 127.0.0.1 (127.0.0.1).
220 ProFTPD Server (ProFTPD Default Installation) [127.0.0.1]
Name (127.0.0.1:root): student
331 Password required for student
Password:
530 Login incorrect.
Login failed. ←ログイン拒絶
```

```
Remote system type is UNIX.  
Using binary mode to transfer files.  
ftp> quit  
221 Goodbye.
```

第 8 章

8.1.3 練習

```

1.
[root@h006 ~]# less /etc/rsyslog.conf
2.
[root@h006 ~]# tty
/dev/pts/0
[root@h006 ~]# vi /etc/rsyslog.conf
[root@h006 ~]# grep authpriv /etc/rsyslog.conf
*.info;mail.none;authpriv.none;cron.none                /var/log/messages
# The authpriv file has restricted access.
#authpriv.*                                              /var/log/secure
authpriv.*                                               /dev/pts/0
3.
[root@h006 ~]# systemctl restart rsyslog
[root@h006 ~]# Aug 19 08:48:20 h006 polkitd[630]: Unregistered Authentication Agent
for unix-process:30466:583543 (system bus name :1.34, object path
/org/freedesktop/PolicyKit1/AuthenticationAgent, locale ja_JP.UTF-8) (disconnected
from bus)
4.
[root@h006 ~]# tail /var/log/messages
Aug 19 08:48:01 h006 rsyslogd: [origin software="rsyslogd" swVersion="8.24.0" x-
pid="914" x-info="http://www.rsyslog.com"] rsyslogd was HUPed
Aug 19 08:48:20 h006 systemd: Stopping System Logging Service...
Aug 19 08:48:20 h006 rsyslogd: [origin software="rsyslogd" swVersion="8.24.0" x-
pid="914" x-info="http://www.rsyslog.com"] exiting on signal 15.
Aug 19 08:48:20 h006 systemd: Starting System Logging Service...
Aug 19 08:48:20 h006 rsyslogd: [origin software="rsyslogd" swVersion="8.24.0" x-
pid="30476" x-info="http://www.rsyslog.com"] start
Aug 19 08:48:20 h006 systemd: Started System Logging Service.
Aug 19 08:48:22 h006 systemd: Removed slice User Slice of root.
Aug 19 08:48:22 h006 systemd: Stopping User Slice of root.
5.
[root@h006 ~]# useradd logtest
Aug 19 08:49:19 h006 useradd[31698]: new group: name=logtest, GID=2008
Aug 19 08:49:19 h006 useradd[31698]: new user: name=logtest, UID=2007, GID=2008,
home=/home/logtest, shell=/bin/bash
6.
[root@h006 ~]# vi /etc/rsyslog.conf
[root@h006 ~]# systemctl restart rsyslog

```

8.2.2 練習

```

[student@h006 ~]$ at now +1 min
at> date | mail -s "at test" student

```

```

at> ^D
job 2 at Sun Aug 19 09:09:00 2018
[student@h006 ~]$ date
2018年  8月 19日 日曜日 09:08:34 JST
[student@h006 ~]$ at -l
2    Sun Aug 19 09:09:00 2018 a student
[student@h006 ~]$
新しいメールが /var/spool/mail/student にあります
[student@h006 ~]$ tail /var/spool/mail/student
Subject: at test
User-Agent: Heirloom mailx 12.5 7/5/10
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit
Message-Id: <20180819000900.71AB712127B@h006.s121.la.net>
From: student@h006.s121.la.net (LA Standard)

2018年  8月 19日 日曜日 09:09:00 JST

```

8.3.3 練習

```

1.
[root@h006 ~]# yum install ntp
(省略)
Downloading packages:
(1/3): ntpdate-4.2.6p5-28.el7.centos.x86_64.rpm           |  86 kB   00:00
(2/3): autogen-libopts-5.18-5.el7.x86_64.rpm            |   66 kB   00:00
(3/3): ntp-4.2.6p5-28.el7.centos.x86_64.rpm             |  549 kB   00:00
-----
合計                               1.3 MB/s | 701 kB  00:00
(省略)
完了しました!
2.
[root@h006 ~]# date
2018年  8月 19日 日曜日 09:20:04 JST
3.
[root@h006 ~]# ntpdate 0.jp.pool.ntp.org 1.jp.pool.ntp.org 2.jp.pool.ntp.org
19 Aug 10:02:38 ntpdate[32072]: step time server 133.243.238.243 offset 2403.085328
sec
4.
[root@h006 ~]# date
2018年  8月 19日 日曜日 10:02:45 JST
5.
[root@h006 ~]# crontab -e
crontab: installing new crontab
[root@h006 ~]# crontab -l
10 10 * * * /usr/sbin/ntpdate -s 0.jp.pool.ntp.org 1.jp.pool.ntp.org 2.jp.pool.ntp.org

```

```

10:10頃の /var/log/message
Aug 19 10:10:06 h006 ntpdate[32128]: 139.162.66.25 rate limit response from server.
Aug 19 10:10:13 h006 ntpdate[32128]: adjust time server 133.243.238.243 offset
-0.000005 sec

```

3.4.4 練習

1.

```
[student@h006 ~]$ env LANG=C man 5 passwd
```

```

name:password:UID:GID:GECOS:directory:shell

The field are as follows:

name      This is the user's login name.  It should not contain capi-
          tal letters.

password  This is either the encrypted user password, an asterisk
          (*), or the letter 'x'.  (See pwconv(8) for an explanation
          of 'x'.)

UID       The privileged root login account (superuser) has the user
          ID 0.

GID       This is the numeric primary group ID for this user.  (Addi-
          tional groups for the user are defined in the system group
          file; see group(5)).

GECOS    This field (sometimes called the "comment field") is
Manual page passwd(5) line 43/108 54% (press h for help or q to quit)

```

2.

```
[student@h006 ~]$ whatis crypt
```

```
crypt: 適切なものではありませんでした。
```

```
[student@h006 ~]$ apropos crypt
```

```

cmsutil (1)          - Performs basic cryptograpic operations, such as encryp...
crypttab (5)         - Configuration for encrypted block devices
des_modes (7ssl)     - the variants of DES and other crypto algorithms of Ope...
Filter::decrypt (3pm) - template for a decrypt source filter
gpg (1)              - OpenPGP encryption and signing tool
gpg-zip (1)          - Encrypt or sign files into an archive
gpg2 (1)             - OpenPGP encryption and signing tool
shadow (3)           - encrypted password file routines
symcryptrun (1)      - Call a simple symmetric encryption tool
systemd-cryptsetup (8) - Full disk decryption logic
systemd-cryptsetup-generator (8) - Unit generator for /etc/crypttab
systemd-cryptsetup@.service (8) - Full disk decryption logic

```

第9章

9.1.2 練習と補足

```
[student@h010 ~]$ cat /proc/swaps
Filename                                Type              Size      Used      Priority
/dev/dm-1                               partition        991228    0         -1

[student@h010 ~]$ swapon
NAME      TYPE      SIZE USED PRIO
/dev/dm-1 partition 968M  0B   -1
```

swapon はスワップ領域を追加するコマンドですが、引数なしで現状を表示します。

9.1.4 練習

```
1.
[student@h010 ~]$ df -h /home
ファイルシステム      サイズ  使用  残り  使用%  マウント位置
/dev/mapper/centos-home 175G 132M 174G   1% /home

[student@h010 ~]$ df -i /home
ファイルシステム      Iノード I使用  I残り  I使用%  マウント位置
/dev/mapper/centos-home 91318272 2717 91315555   1% /home

2.
[student@h010 ~]$ du -sh ~/
99M   /home/student/
```

9.2.2 練習

```
[root@h006 ~]# pvs
PV      VG      Fmt  Attr  PSize  PFree
/dev/sda2 centos lvm2 a-- 66.96g 0

[root@h006 ~]# vgs
VG      #PV #LV #SN Attr   VSize  VFree
centos  1  3  0 wz--n- 66.96g 0

[root@h006 ~]# lvs
LV      VG      Attr   LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
home   centos -wi-ao---- <21.66g
root   centos -wi-ao---- <44.36g
swap   centos -wi-ao---- 976.00m
```

各コマンドはそれぞれ、PV、VG、LVの概要を表示します。より詳しい内容は pvdisplay, vgdisplay, lvdisplay で確認できます。

9.3.3 練習

```
1.
[root@h006 ~]# xfsdump -l 0 -L "boot backup" -M "disk" -f /var/boot.bkup /dev/sda1
xfsdump: using file dump (drive_simple) strategy
(省略)
```

```
xfsdump: stream 0 /var/boot.bkup OK (success)
xfsdump: Dump Status: SUCCESS
2.
[root@h006 ~]# cp /etc/hosts /boot/
[root@h006 ~]# xfsdump -l 1 -L "boot backup 1" -M "disk" -f /var/boot1.bkup /dev/sda1
xfsdump: using file dump (drive_simple) strategy
(省略)
xfsdump: stream 0 /var/boot1.bkup OK (success)
xfsdump: Dump Status: SUCCESS
[root@h006 ~]# ls -l /var/boot*
-rw-r--r-- 1 root root 215942192  8月 23 09:45 /var/boot.bkup
-rw-r--r-- 1 root root  23448  8月 23 09:46 /var/boot1.bkup
3.
[root@h006 ~]# cd ~
[root@h006 ~]# xfsrestore -i -f /var/boot.bkup .
xfsrestore: using file dump (drive_simple) strategy
(省略)
-> ls
          89 initramfs-3.10.0-862.11.6.el7.x86_64.img
          (省略)
          74 vmlinuz-3.10.0-862.el7.x86_64
-> add symvers-3.10.0-862.el7.x86_64.gz
-> extract
----- end dialog -----
xfsrestore: restoring non-directory files
xfsrestore: restore complete: 18 seconds elapsed
xfsrestore: Restore Summary:
xfsrestore: stream 0 /var/boot.bkup OK (success)
xfsrestore: Restore Status: SUCCESS
[root@h006 ~]# ls
anaconda-ks.cfg          symvers-3.10.0-862.el7.x86_64.gz
initial-setup-ks.cfg    typescript
```

第 10 章

10.2.2 練習

1.

```
[root@h006 ~]# shutdown -r now
```

カーネル選択画面で、[e]を押す

```
CentOS Linux (3.10.0-862.11.6.el7.x86_64) 7 (Core)
CentOS Linux (3.10.0-862.9.1.el7.x86_64) 7 (Core)
CentOS Linux (3.10.0-862.el7.x86_64) 7 (Core)
CentOS Linux (0-rescue-b8b8b427fe684cce853021a636d9a9b5) 7 (Core)

Use the ↑ and ↓ keys to change the selection.
Press 'e' to edit the selected item, or 'c' for a command prompt.
The selected entry will be started automatically in 5s.
```

linux16 で始まる行の最後に「rw init=/bin/sh」を追加し、^X

```
insmod xfs
set root='hd0,msdos1'
if [ x${feature_platform_search_hint} = xy 1; then
    search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hin\
t-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1' d003f429-c\
3c9-4120-8cb0-88d54304039e
else
    search --no-floppy --fs-uuid --set=root d003f429-c3c9-4120-8cb0-88d5\
4304039e
fi
linux16 /vmlinuz-3.10.0-862.11.6.el7.x86_64 root=/dev/mapper/centos_h0\
06-root ro rd.lvm.lv=centos_h006/root rd.lvm.lv=centos_h006/swap rhgb quiet LA\
NG=ja_JP.UTF-8 rw init=/bin/sh
initrd16 /initramfs-3.10.0-862.11.6.el7.x86_64.img

Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
discard edits and return to the menu. Pressing Tab lists
possible completions.
```

2.

```
sh-4.2# id          ←パスワード入力をスキップしていきなりスーパーユーザーになっている。
```

```
uid=0(root) gid=0(root) groups=0(root)
```

3.

```
sh-4.2# exec /sbin/init
```

4.

```
[root@h006 ~]# grub2-setpassword
```

```
Enter password:
```

```
Confirm password:
```

```
[root@h006 ~]# cat /boot/grub2/user.cfg
```

```
GRUB2_PASSWORD=grub.pbkdf2.sha512.10000.B571CBE7766AC005BCA32B370AC7711F3049EC4F3E695D
E9444C79F45790D86A24A9C57957E34291F87BEC730809DB8334C3E0456EFD7DAA443B0E08492A6EF2.00C
69513737A8DDB2054AF6386380419AC6EAEF8382A3E9C57A0CC94F980E8B1F1D1FFFA7039E061EFB968F52
```

```
EF00D949E8C63032B89555998F918C3A8F6A980
5.
    (コンソールにてカーネル選択で [e] を押した直後)
Enter username:
root
Enter password:
*****
    (ブートパラメータ編集では特に変更せず、^X でそのまま起動)
6.
[root@h006 ~]# rm /boot/grub2/user.cfg
```

10.2.4 練習

割愛

10.2.5 練習

```
1.
[root@h006 ~]# ps aux
USER      PID %CPU %MEM  VSZ   RSS TTY      STAT START   TIME COMMAND
root      1  0.0  0.0 191188 4116 ?        Ss   10:29   0:01
/usr/lib/systemd/systemd --switched-root --system --deserialize 22
root      2  0.0  0.0    0     0 ?        S    10:29   0:00 [kthreadd]
root      3  0.0  0.0    0     0 ?        S    10:29   0:00 [ksoftirqd/0]
root      4  0.0  0.0    0     0 ?        S    10:29   0:00 [kworker/0:0]
    (省略)
[root@h006 ~]# ss -antup
Netid  State      Recv-Q Send-Q Local Address:Port          Peer Address:Port
udp    UNCONN    0      0      127.0.0.1:323              *: *
      users: (("chronyd", pid=882, fd=1))
udp    UNCONN    0      0      *:983                      *: *
      users:
      (("rpcbind", pid=809, fd=7))
    (省略)
2.
[root@h006 ~]# systemctl stop cups
Warning: Stopping cups.service, but it can still be activated by:
  cups.path
  cups.socket
[root@h006 ~]# systemctl status cups
● cups.service - CUPS Printing Service
   Loaded: loaded (/usr/lib/systemd/system/cups.service; enabled; vendor preset:
   enabled)
   Active: inactive (dead) since 木 2018-08-23 11:06:05 JST; 13s ago
    (省略)
3.
[root@h006 ~]# systemctl list-unit-files
UNIT FILE                                     STATE
```

```
proc-sys-fs-binfmt_misc.automount          static
(省略)

4.
[root@h006 ~]# systemctl disable cups
Removed symlink /etc/systemd/system/multi-user.target.wants/cups.path.
Removed symlink /etc/systemd/system/multi-user.target.wants/cups.service.
Removed symlink /etc/systemd/system/sockets.target.wants/cups.socket.
Removed symlink /etc/systemd/system/printer.target.wants/cups.service.

5.
[root@h006 ~]# reboot
[root@h006 student]# systemctl list-unit-files | grep cup
cups.path                                disabled
cups-browsed.service                    disabled
cups.service                            disabled
cups.socket                              disabled
```

10.4.3 練習

```
1.
[root@h006 ~]# iptables -P INPUT DROP

2.
[root@h006 ~]# iptables -P FORWARD DROP

3.
[root@h006 ~]# iptables -F
[root@h006 ~]# iptables -L
Chain INPUT (policy DROP)
target    prot opt source                destination
Chain FORWARD (policy DROP)
target    prot opt source                destination
Chain OUTPUT (policy ACCEPT)
target    prot opt source                destination
ブラウザで指定した URL が表示されない

4.
[root@h006 ~]# ping www.google.com
ping: www.google.com: 名前またはサービスが不明です

5.
[root@h006 ~]# ssh 10.20.121.10
ssh: connect to host 10.20.121.10 port 22: Connection timed out

6.
[root@h006 ~]# iptables -P INPUT ACCEPT
```

10.4.4 練習

```
1.
[root@h006 ~]# iptables -A INPUT -s 10.20.121.10 -j DROP

2.
[root@h006 ~]# iptables -L
```

```
Chain INPUT (policy ACCEPT)
target    prot opt source                destination
DROP     all  --  h010.s121.la.net      anywhere

Chain FORWARD (policy DROP)
target    prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target    prot opt source                destination

3.
[student@h010 ~]$ ssh 10.20.121.6
ssh: connect to host 10.20.121.6 port 22: Connection timed out
```

10.5 確認問題

自ホストが 0.20.121.6 の場合

```
[root@h006 ~]# iptables -F                                ※一旦設定を初期化
[root@h006 ~]# iptables -P INPUT DROP
[root@h006 ~]# iptables -A INPUT -p all -d localhost -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p icmp -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p tcp -d 10.20.121.6 --dport 80 -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p tcp -d 10.20.121.6 --dport 53 -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p udp -d 10.20.121.6 --dport 53 -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p tcp -d 10.20.121.6 --dport 25 -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p tcp -d 10.20.121.6 --dport 110 -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p tcp -d 10.20.121.6 --dport 22 -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p tcp -d 10.20.121.6 --sport 80 -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p tcp -d 10.20.121.6 --sport 53 -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p udp -d 10.20.121.6 --sport 53 -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p tcp -d 10.20.121.6 --sport 25 -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p tcp -d 10.20.121.6 --sport 110 -j ACCEPT
[root@h006 ~]# iptables -A INPUT -p tcp -d 10.20.121.6 --sport 22 -j ACCEPT

[root@h006 ~]# iptables -L
Chain INPUT (policy DROP)
target    prot opt source                destination
ACCEPT    all  --  anywhere              localhost
ACCEPT    all  --  anywhere              localhost
ACCEPT    tcp  --  anywhere              10.20.121.6          tcp dpt:http
ACCEPT    tcp  --  anywhere              10.20.121.6          tcp dpt:domain
ACCEPT    udp  --  anywhere              10.20.121.6          udp dpt:domain
ACCEPT    tcp  --  anywhere              10.20.121.6          tcp dpt:smtp
ACCEPT    tcp  --  anywhere              10.20.121.6          tcp dpt:pop3
ACCEPT    tcp  --  anywhere              10.20.121.6          tcp dpt:ssh
ACCEPT    tcp  --  anywhere              10.20.121.6          tcp spt:http
ACCEPT    tcp  --  anywhere              10.20.121.6          tcp spt:domain
```

```
ACCEPT    udp  --  anywhere          10.20.121.6      udp spt:domain
ACCEPT    tcp  --  anywhere          10.20.121.6      tcp spt:smtp
ACCEPT    tcp  --  anywhere          10.20.121.6      tcp spt:pop3
ACCEPT    tcp  --  anywhere          10.20.121.6      tcp spt:ssh
```

```
Chain FORWARD (policy DROP)
```

```
target    prot opt source          destination
```

```
Chain OUTPUT (policy ACCEPT)
```

```
target    prot opt source          destination
```

2. FORWARD チェインの設定

```
# iptables -P FORWARD DROP
```

3. OUTPUT チェインの設定

```
# iptables -P OUTPUT ACCEPT
```