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事が一般的です。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をはらんでいました。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情報などが第三者に渡るとサーバーが乗っ取られる危険があります。

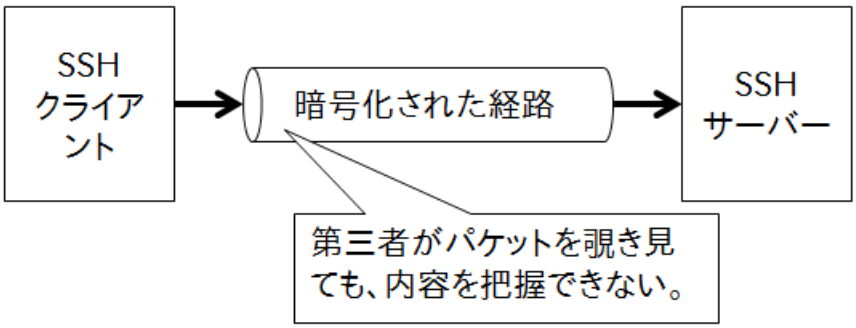
そこで、安全な通信を実現するために次の2つの要件が必要となります。

・盗聴されてもデータが読み取られないこと
・確実なユーザーとホストの認証

この要請に応える技術が「暗号化」です。暗号化した通信によりデータをそのままでは読み取れなくできます。その結果、認証関係の情報も安心して送信できるようになり、上記の2つの要請を満たすことができます。今回は、リモートアクセスに際して有効な暗号化を提供する SSH について見ていくことにします。

1.1.2 SSH プロトコル

TELNET を用いたリモートログインでは、全てのデータが暗号化されないまま送信されます。このため、重要なアカウント情報が盗聴されやすいという危険性があります。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）というプロトコルが開発されました。



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。ログイン先のホストに SSH サーバーが、接続元のホストに SSH クライアントが動作することにより、安全なリモートログインが可能になります。ログインが終了すると、TELNET を用いたリモート操作と同様に、あたかも接続先のホストに直接ログインしたかのように端末を利用することができます。

1.1.3 OpenSSH

SSH は、1995 年当時ヘルシンキ工科大学の研究員だった 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。また、上記以外にも GUI 環境で使用するためのパッケージがいくつかインストールされていることがあります。

［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

1.1.4 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とおりです。

```
ssh　ユーザー名@接続先のホスト
```

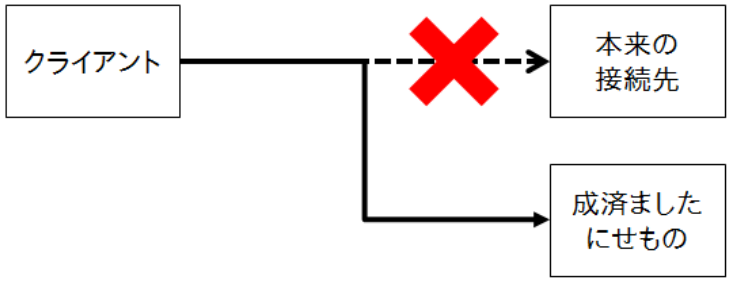
接続先のホストをホスト名または IP アドレスで指定します。サーバー側ホストのログインユーザーを指定するためには「[ユーザー名]@」を記述します。省略するとクライアント側カレントユーザー名と同名のサーバー側ユーザーとしてログインすることになります。

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

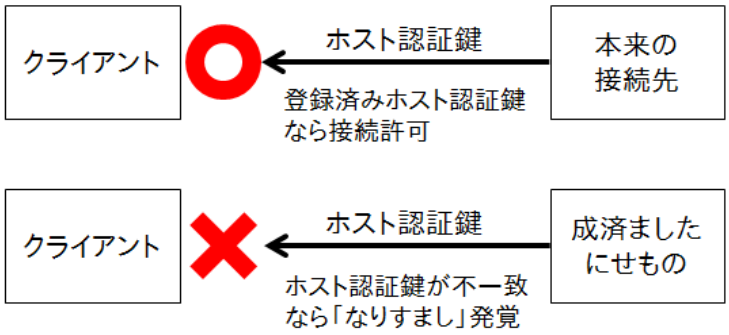
```
$　ssh　student@10.20.142.6
```

1.1.5 ホスト間認証

通信を暗号化しても、全てが安全になるというわけではありません。攻撃者のとる手法として、「なりすまし」があります。接続先のホストになりすまし、重要なデータを奪うというものです。



このなりすましからクライアントを守るためには、ログインしているコンピューターが意図しているホストかどうかを確認する必要があります。SSH では通信の開始に SSH サーバー側からクライアントに対してホスト間認証用の公開鍵と呼ばれるデータが送信されます。この公開鍵は、固有のペアの鍵となる秘密鍵を持ち、攻撃者がなりすましたサーバーの場合、正規のサーバーと同じペア鍵のもう一方を持っていないという事実から、その攻撃を認識することができます。



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/YA8o8rOtD/LoyxbA1NFzg2WRG6ZUB59CJ69No7ooBB2bvMVX3pmtSK4E
NrEbJNI=
```

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。

1.1.6 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:10.20.142.6 へのログイン(2回目以降)

```
$ ssh   student@10.20.142.6
student@10.20.142.6's password: [パスワードを入力]
Last login: Wed May  6 22:50:10 2009 from h100.s16.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とパスワード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. ssh コマンドを用いて、講師のマシンにログインします。このとき用いるユーザーアカウントについては講師の指示に従って下さい。

2. ログイン先で、コマンド「ip show addr」を使いネットワーク設定を確認し、他のマシンにログインしている事を確認します。

3. w コマンドを用いると、現在システムにログインしているユーザーの情報を表示させることができます。得られた表示のうち、「FROM」欄に接続元が記述されています。ログイン先で w コマンドを実行して、自分の接続を示している行を確認します。

4. exit コマンド、または Ctrl ＋ d を利用して、ログアウトします。

5. 自分のマシン内の ~/.ssh/known_hosts を表示し、接続したホストの認証鍵が登録されていることを確認します。

今回はログインするために UNIX パスワード認証を用いましたが、暗号化しているとはいえ、システムのアカウント情報がネットワーク上を流れます。より高度な安全性を実現するために、公開鍵暗号認証と呼ばれる認証方法があります。この方法は次章で解説します。

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 または service コマンドを用います。

```
    # systemctl  start  sshd
         または
    # service  sshd  start
```

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)
   Active: active (running) since 火 2015-07-28 08:25:36 JST; 34min ago
 Main PID: 1072 (sshd)
   CGroup: /system.slice/sshd.service
           └ 1072 /usr/sbin/sshd -D
```

［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. 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。あわせて、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。あわせて、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.

# This sshd was compiled with PATH=/usr/local/bin:/usr/bin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value wher
# possible, but leave them commented.  Uncommented options override the
# default value.

# 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多数見られますが、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 である事を意味しています。

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。 SSH2 のみを許可する事が推奨されている。（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_key, /etc/ssh /ssh_host_dsa_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]。この時間を過ぎても放置、されると自動的に接続が切断される。（2m）
PermitRootLogin	ssh を利用したユーザー root による直接ログイン可否。 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。セキュリティレベルを向上のため、「許可しない」が増えている。（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 アカウント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 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 アカウントでのログインを試みます。
7. 設定を元に戻し、sshd の再起動を行います。

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。

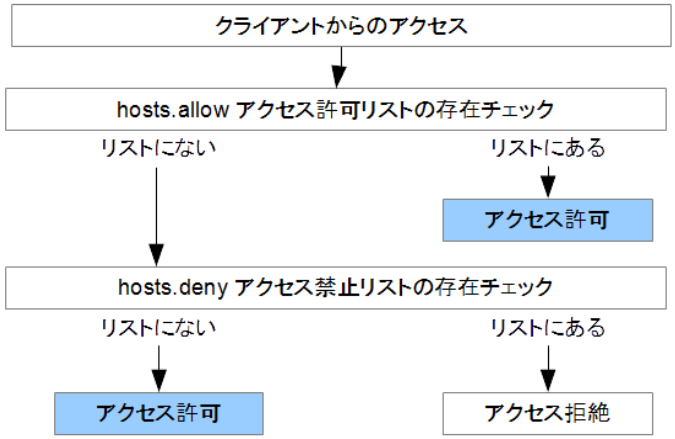
デフォルトの hosts.allow ファイルと hosts.deny ファイルには、何の設定も記述されておらず、全てのホストからの全てのサービスへの接続が許可されています。

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、それ以外は不許可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事に加え、「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 TCPwrapper（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、「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、ダウンロードして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、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代わりはありません。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

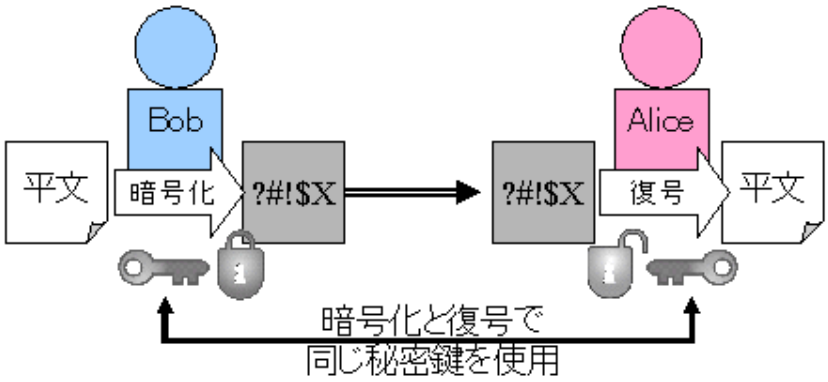
暗号化には大きく分けて2種類あります。暗号化すると二度と元に戻せない「片方向暗号（ダイジェスト）」と、鍵とよばれる情報を使って元に戻す「鍵暗号」です。片方向暗号では「暗号化された情報が同じであれば、変換前のデータも同じ」という前提を用いたもので、主に認証の際に用いられます。後者は、データを暗号化したのち、データを再変換して元のデータに戻（復号）します。

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い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

データの暗号化がアルゴリズムと鍵から生成されることは前項のとおりです。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。



共通鍵方式では、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事になります。

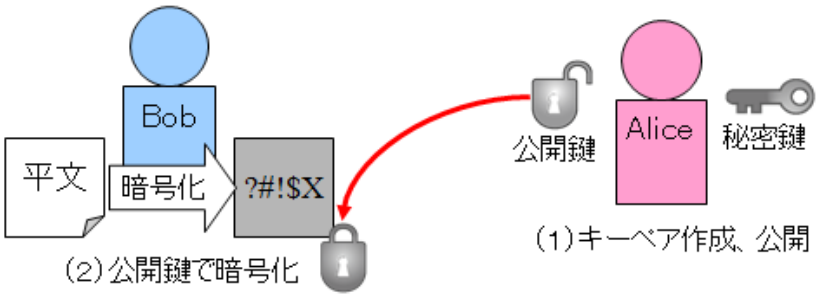
これは「鍵配送問題」と呼ばれ、暗号の歴史の中で非常に多くの人を悩ませてきました。

この「鍵配送問題」を解決したのが、「公開鍵暗号(Public key domain)」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つを用います。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。

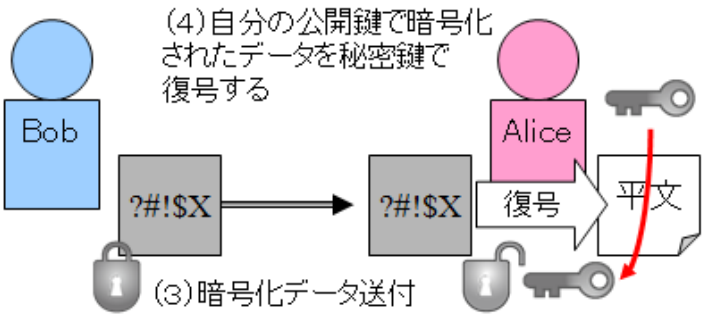
1. 受け取り側の Alice は、公開鍵暗号で必要となるキーペア（公開鍵、秘密鍵）を用意し、公開鍵を誰でもアクセスできる場所に置きます。

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. 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。
2. SSH クライアントで SSH サーバーにアクセスする。
3. 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。
4. 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。
5. 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には以下のとおりです。

プロトコル	SSH Ver.1	SSH Ver.2
アルゴリズム	RSA	RSA, DSA, 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@localhost.localdomain
The key's randomart image is:
+--[ DSA 1024]----+
|++o      ..      |
| ...   ...o      |
|..o   o  E.      |
|.o . . + ..      |
|. 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/ZTWCmGAXZLQi4P3RzIkQc6Jor4zFjz5yyZPO1hh
rWVv7VsuXN+qILsWQOajnXFdibhQcaW/dGF8vldBIDfN2lq5y94/hhLHBJZsEreC
3p86XD8hbWkC+zz2Gtt4bxR05cz3Xwakcs5R6cVGDcxmSkkBh84EFu5Taq6fqpdM
q3yXdZZL0WZlYD801IWZFZsNYU1jiLBElWOon3ZOAbBargfU7bLOFZKTRFvKoT3V
fhXs9SD9DlkH+AxYwu+w3456Es9f/B6LB8CQDho/xQrzUplDdKOcXtw4kpqqII7O
28eHxRUbmnYlFl3H/fPH0bhzZXcAdo3iKTo42oDFEwDuuo7zE02FyOkm/hTalb3L
a+GwNH0vSHjV6GTVBXzyunV0Pc6kVH1WI6/Ik5Kjca4I5KIEoae4ZtAUr10diOYo
L+U0u9jbL3yULipQsvQgFblaL+1jeWVOuRgZ20Ie3UThCjr3alieuDvZIfp3wgi+
0uA7Jfbro0EEtNkInMpJ6Zu0McW3Lrkl3qbIAG+OXkngc2kMkeQCDtH2Unv5yRGR
gUozlmcfwVGgW+2WfqdHRg==
```

id_dsa.pub（公開鍵）の例

```
ssh-dss AAAAB3NzaC1kc3MAAACBALn2Gc7zddXGzRvxQiXO7HtNTjC/oWiLUUz+hQ0mMo
Gnzo4KQSdmqHW0Jaayf6UcM1tPeC+G1nQ6VWb9LDQCh2FavBJGROjP+GM6Dq42rpzH5cmNNLN4/XfRcTx7
O39bqXRh8K7rXhjBnkHhWG+hcz+6PBCdMJVE2axgu/8gP+fpAAAAFQCw2ZveluorBL5GIhLrXIn2dqXLqQ
AAAIBl2cMbAoU9GpN6ETohFmAWMMO1AbnV3YkUsrIkIVhCOwvBZbOUCmdxazhdQo4uB8KgvnwNQq5UdiZX
kv61XexK92R0cDsNUs3M4GKGIEP4L1h8J6N5Cfv/EkrPwSImDQLeGhxSDJQ44HCG6D0Z6xc8HeAbLEFCTi
uTFAvbkz2QCwAAAIEAiU5UYy7sthZdMcuEjQUzRE7kM/QKpGIJQv0l6UbIJ5FmaOZI6Apc/o+bXeZndSPi
W9iNsEeG7Th/H3X3SajuEV2PKtipnEB6gBemlJsmITD+bbEJCHEvyICPwCpEoS1x5Qt00lIFEI+OmbjbZE
nlrfMc50v95/ms9igZzfOVm+E= student@localhost.localdomain
```

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、鍵生成の際の鍵のアルゴリズムとして、「-t rsa」とオプション指定します。SSH1 でも RSA 認証鍵を用いることはできますが、この場合は「-t rsa1」と指定します。

生成されるアルゴリズムと鍵ファイルは次の通りです。

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

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・USBメモリなどオフライン媒体を用いる事もあります。

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
```

上記の様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、すで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」などに変更する必要があります。

2.2.3 公開鍵認証を用いた SSH サーバーへのログイン

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 2009
[student@h006 ~]$
```

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で構いません。
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を考えます。
4. chmod コマンドを用いて、authorized_keys ファイルのパーミッションを 3.で考えたものに変更します。
5. 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）
RSAAuthentication	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）
PubKeyAuthentication	SSH2 の DSA 認証または RSA 認証を許可する。（yes）
AuthorizedKeysFile	SSH サーバーが参照するユーザーの公開バスをホームディレクトリからの相対バスで指定する。（.ssh/authorized_keys）

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、ローカルのファイルとリモートのファイル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. 隣の方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 は対話式で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ると、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]	ファイルのダウンロード。第 2 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 の修了
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プロセス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内に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内にある 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、この中に 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 PostPet、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。

メールを書き終えると、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、自分で相手のポスト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 RFC821 として制定された後、2001 年 RFC2821・2008 年 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、郵便受けに相当）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 メールの受信

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 からはじまり、1000 番代 2000 番代などを多数経て、2007 年 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。

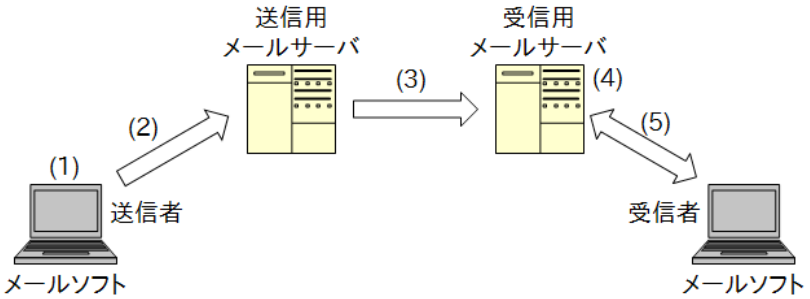
IMAP はサーバー側で処理を行うため、サーバーの性能やディスク容量を多く消費するといったデメリットはありますが、メールがサーバーで一元管理されているため 複数のクライアントからアクセス可能、セキュリティの向上や情報漏えいに強いといったメリットもあります。

IMAP は最初 RFC 1730、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。

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

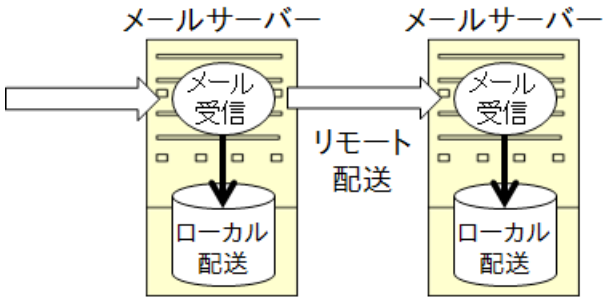
メールの配送の仕組みを整理すると次のようになっています。



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えば、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。

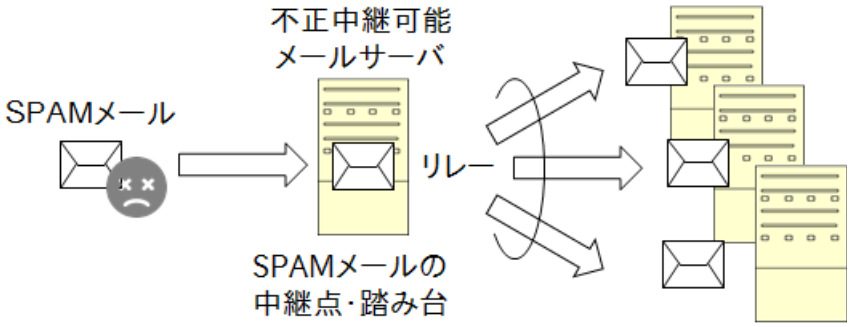
配送は大きく2つのケースがあります。1つはサーバー内にメールボックスがあり、そのメールボックスにメールを格納する「ローカル配送」。もう1つは他のメールサーバーにメールボックスがあるため、そのメールサーバーに転送する「リモート配送」です。



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。この様に誰にでもリレー（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事になります。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付、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存在すれば、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@la.net」という表記を用います。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 = 10 ms.linuxacademy.ne.jp.

Authoritative answers can be found from:
ms.linuxacademy.ne.jp   internet address = 203.174.70.247
```

この例では、linuxacademy.ne.jp ドメインあてのメールは ms.linuxacadmey.ne.jp というサーバーが処理していることを示しています。

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@examp le.net から teacher@la.net 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 や IMAP サーバーが用意されています。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」と呼ばれることになります。

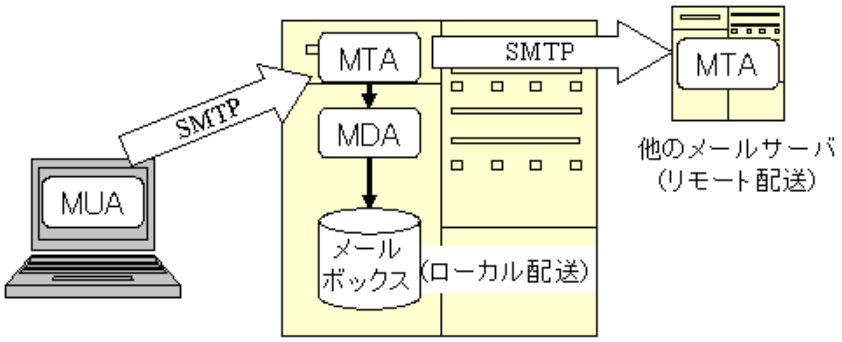
メールクライアントは SMTP/POP/IMAP といったサーバーとのやり取り以外に、アドレス帳や定型文書（テンプレート）といった機能を持つものが多く存在します。

3.2.5 メールサーバーの構成要素

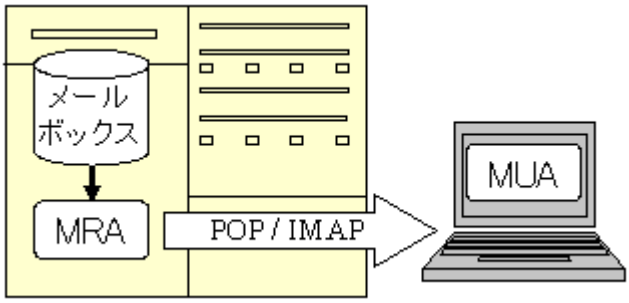
ここまでに解説したメールに関わる機能を整理すると以下の様になります。

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)	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。 関連し、ユーザー認証を行う。

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が表示しています。

	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.s16.la.net  Fri May  8 11:44:19 2009
Return-Path: <student@h231.s16.la.net>
Date: Fri, 8 May 2009 11:44:19 +0900
From: student <student@h231.s16.la.net>
Message-Id: <200905080244.n482iJiQ028392@h231.s16.la.net>
To: teacher@h100.s16.la.net
Subject: Question about Mail System

先生
受講生の○○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

メールヘッダ	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。

3.4 電子メールの配送に利用されるプロトコル

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.s16.la.net ESMTP Sendmail 8.13.8/8.13.8; Fri, 8 May 2009 12:25:16 +0900
EHLO h100.s16.la.net
250-h231.s16.la.net Hello h231.s16.la.net [172.16.1.231], pleased to meet you
250-ENHANCEDSTATUSCODES
    （中略）
250 HELP
MAIL FROM: root@h100.s16.la.net
250 2.1.0 root<root@h100.s16.la.net>... Sender ok
RCPT TO: student@h231.s16.la.net
250 2.1.5 student@h231.s16.la.net<student@h231.s16.la.net>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Date: Fri, 8 May 2009 11:44:19 +0900
From: root <root@h100.s16.la.net>
Message-Id: <200905080244.n482iJiQ028392@h100.s16.la.net>
To: student@h231.s16.la.net
Subject: This is Test Mail

テストメールです!
.
250 2.0.0 n483PGAT029550 Message accepted for delivery
QUIT
221 2.0.0 h231.s16.la.net closing connection
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	メールの本文の始まりを表わす。 「.」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 POP（Post Office Protocol）が利用されます。POP 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 1 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]	ユーザー認証の為のユーザー名
PASS　[パスワード]	USER で指定したもの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

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 などいくつも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 でインストールしてください。

```
    # yum   install   postfix
```

Postfix の設定ファイル 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、すで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］

1. 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。

2. postconf コマンドを利用して、各設定項目に対する設定値を確認します。

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受付可能になる。
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指定されている部分のみ出力します。

```
# 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
          （省略）
```

［練習］

1. main.cf を編集し、次の値を指定します。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  =  h006.s142.la.net
mydomain  =  s142.la.net
myorigin  =  $myhostname
inet_interfaces  =  all
mydestination  =  $myhostname
mynetworks  =  10.20.142.6, 127.0.0.1
```

2. postconf コマンドに「-n」オプションを指定して、1.で編集した内容が適切に解釈されていることを確認します。

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 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 か、service コマンドを利用します。引数には、start・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
    または
# service postfix start
```

Postfix の停止

```
# systemctl stop postfix
    または
# service postfix stop
```

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)
  Active: active (running) since 火 2015-07-28 08:25:36 JST; 6h ago
 Main PID: 1250 (master)
  CGroup: /system.slice/postfix.service
       ├1250 /usr/libexec/postfix/master -w
       ├1252 qmgr -l -t unix -u
       └5288 pickup -l -t unix -u

 7月 28 08:25:36 localhost.localdomain systemd[1]: Starting Postfix Mail Tra...
 7月 28 08:25:36 localhost.localdomain postfix/master[1250]: daemon started ...
 7月 28 08:25:36 localhost.localdomain 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。Cc（カーボンコピー：写しを送る宛先）の入力は必要がなければ、エンターキーの空打ちで終了させます。

```
# mail student@h006.s142.la.net
Subject: Test mail
This is test mail
.
Cc:
```

この後、Postfix は配送処理を行うことになります。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 2015
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 2015 15:07:39 +0900 (JST)
Date: Tue, 28 Jul 2015 15:07:39 +0900
To: student@ h006.s142.la.net
Subject: Test mail
User-Agent: Heirloom mailx 12.5 7/5/10
Message-Id: <20150728060739.D97BDBA@h006.s142.la.net>
From: root@localhost.localdomain (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.で配送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。

```
# telnet  localhost  25
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
220 h231.s16.la.net ESMTP Postfix
EHLO h006.s142.la.net
250-h006.s142.la.net
    （中略）
250 DSN
MAIL FROM: root
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. 再度、2.を行ってメールが配送されていることを確認します。
5. 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）を必要とする送信要求は、設定項目 mynetworks で設定されたホスト、ネットワークからしか許可されていません。許可されていないホストから中継が必要なメール送信要求を行おうとすると、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、syslog 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、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 localhost postfix/master[1195]: daemon started -- version 2.10.1,
configuration /etc/postfix
Jul 27 19:19:54 localhost postfix/postfix-script[1160]: starting the Postfix mail
system
Jul 27 19:19:54 localhost postfix/master[1162]: daemon started -- version 2.10.1,
configuration /etc/postfix
Jul 28 08:25:36 localhost postfix/postfix-script[1248]: starting the Postfix mail
system
Jul 28 08:25:36 localhost postfix/master[1250]: daemon started -- version 2.10.1,
configuration /etc/postfix
Jul 28 15:07:39 localhost postfix/pickup[6595]: D97BDBA: uid=0 from=<root>
Jul 28 15:07:39 localhost postfix/cleanup[6619]: D97BDBA: messageid=<20150728060
739.D97BDBA@localhost.localdomain>
Jul 28 15:07:39 localhost postfix/qmgr[1252]: D97BDBA: from=<root@localhost.
localdomain>, size=465, nrcpt=1 (queue active)
Jul 28 15:07:39 localhost postfix/local[6621]: D97BDBA: to=<student@localhost.
localdomain>, orig_to=<student@localhost>, 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 localhost postfix/qmgr[1252]: D97BDBA: removed
```

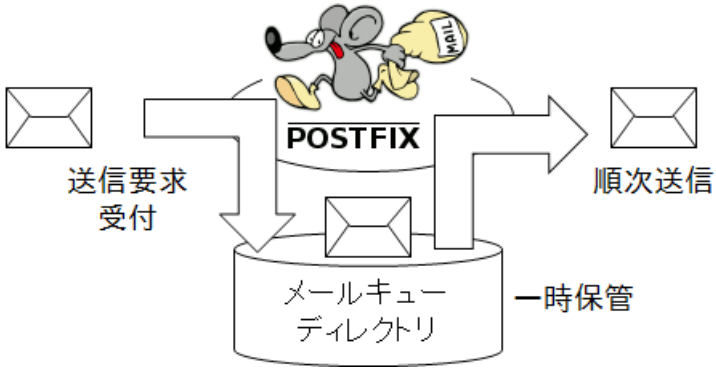
［練習］

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。キューは、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。

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@localhost.localdomain
                      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、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。

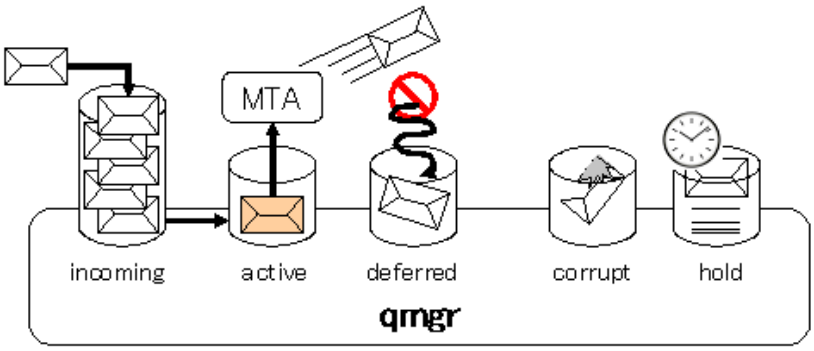
2. root ユーザーで mailq コマンドを実行し、キューにメールが保存されていることを確認します。送信待ちになっているメールの Queue ID を調べます。

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のキューを削除します。

5. mailq コマンドを実行し、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。

```
（抜粋）
# 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（上の3行の例では、最終的には student のメールボックスに配送されます。）、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、実際の配送先を知ります。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 で受け取れ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、システム上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」を配しておきます。（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@h132.s16.la.net
```

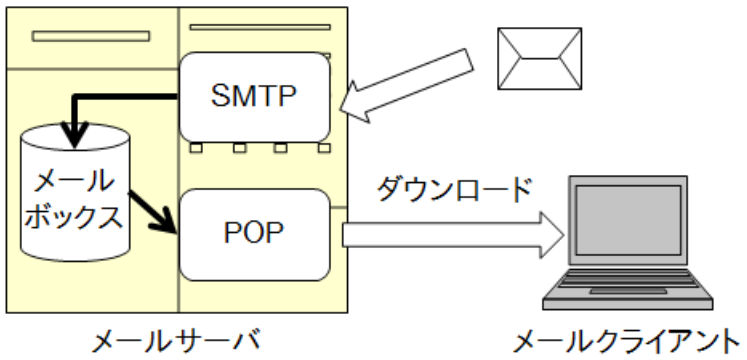
5　メールサーバー（3）

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の1つです。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 はさまざまな 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

メール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ています。

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@localhost.localdomain>
From: root@localhost.localdomain (root)

This is test mail

From root@localhost.localdomain  Tue Jul 28 15:55:03 2015
Return-Path: <root@localhost.localdomain>

Date: Tue, 28 Jul 2015 15:55:03 +0900
To: student@localhost.localdomain
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@localhost.localdomain>
From: root@localhost.localdomain (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 機能を提供するローカル MDA（MRA）という位置づけで、セキュリティ対策や IPv6 に対応したりしていて、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。

```
（抜粋）
#   mail_location = mbox:~/mail:INBOX=/var/mail/%u
#   mail_location = mbox:/var/mail/%d/%1n/%n:INDEX=/var/indexes/%d/%1n/%n
#
mail_location = mbox:~/mail:INBOX=/var/mail/%u                ←追加
# mailboxes, or ln -s /secret/shared/box ~/mail/mybox would allow reading it).
#mail_access_groups =
mail_access_groups = mail                                 ←追加
```

5.2.3 Dovecot の起動

Dovecot の起動、停止は他のサービス同様 systemctl 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 のインターフェイスを持つ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 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.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。

［練習］

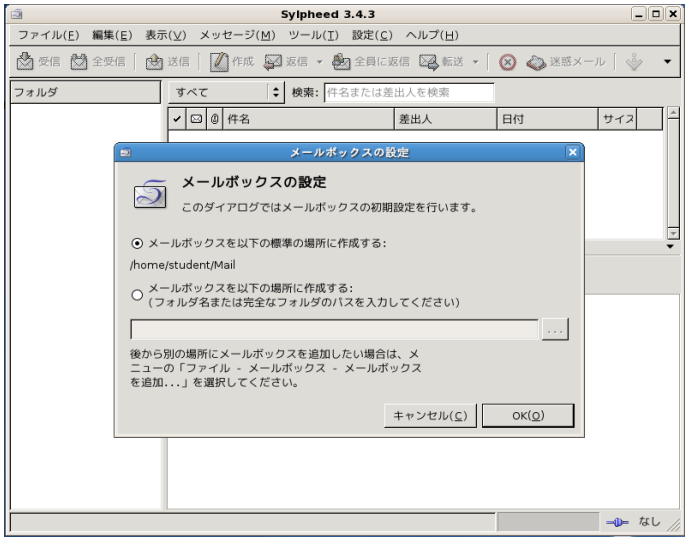
講師の指示に従って Sylpheed のパッケージを入手してインストールを行ってください。

5.3.2 Sylpheed の起動とアカウントの設定

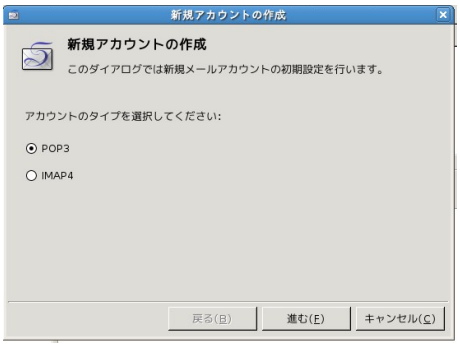
Sylpheed を起動するには、X Window System 上の端末で sylpheed コマンドを実行します。

```
$  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」を選択してもかまいません。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。

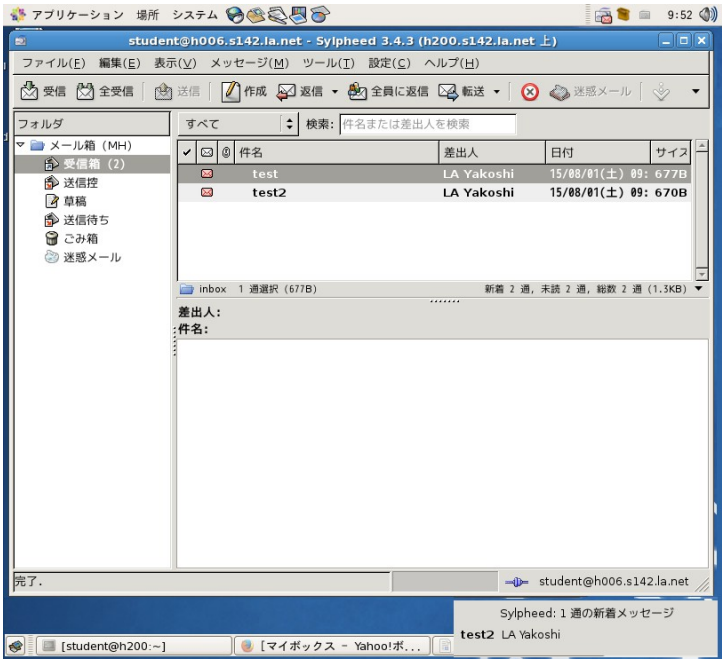


次に受信に使うプロトコルを POP、IMAP の中から選択します。今回は POP を選択します。

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. 上記設定方法を参考に 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。

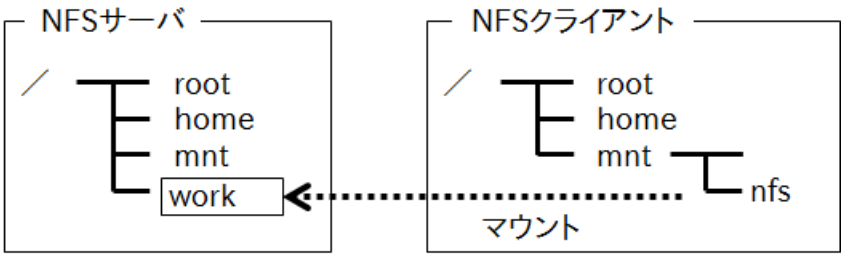
6 NFS

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 クライアントを使って NFS サーバーにアクセスすることにより、マウントすることが可能です。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を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。

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                          static
nfs-config.service                          static
nfs-idmap.service                           static
```

```
nfs-idmapd.service                          static
nfs-lock.service                            static
nfs-mountd.service                          static
nfs-secure-server.service                   static
nfs-secure.service                          static
nfs-server.service                          disabled
nfs-utils.service                           static
nfs.service                                 disabled
```

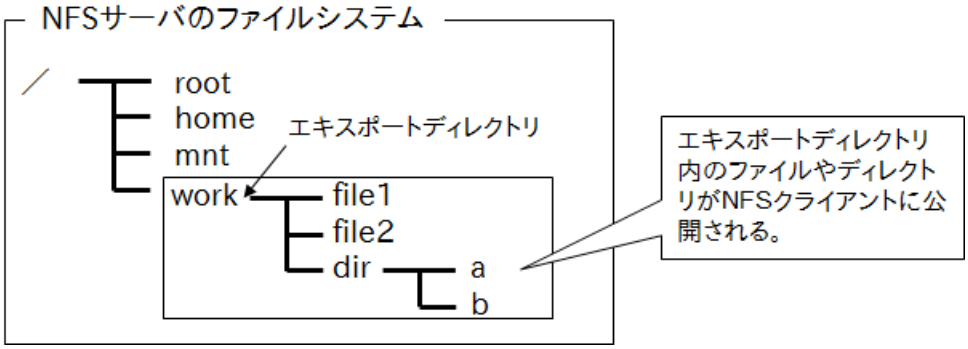
NFS サーバーで、systemctl を使って nfs-server を起動します。クライアントについては特にサービスを起動する必要はありません。

```
# 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)
  Active: active (exited) since 火 2015-07-28 21:41:39 JST; 1min 57s ago
 Process: 11814 ExecStopPost=/usr/sbin/exportfs -f (code=exited, status=0/SUCCESS)
 Process: 11810 ExecStopPost=/usr/sbin/exportfs -au (code=exited,
status=0/SUCCESS)
 Process: 11809 ExecStop=/usr/sbin/rpc.nfsd 0 (code=exited, status=0/SUCCESS)
 Process: 11824 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 マウントの実行

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、サブディレクトリも含めて全て 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 の中で指定します。

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 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 ネットワーク上にエクスポート(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()との間に空白はありません。

6.2.2 エクスポート設定の反映と確認

前節の設定を有効にするためには、exportfs コマンドに「-a」オプションを渡して実行します。なお、これを実行するためには、事前に NFS サービスが実行され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、/export 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」オプションを渡して実行します。

```
# 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)
```

上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 エクスポートディレクトリのマウント

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 コマンドで確認するとよいでしょう。

```
# mount  -t  nfs  h006.s142.la.net:/work  /media/nfs
# 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
h006.s142.la.net:/work      1020672 65664 955008     7% /media/nfs
```

サーバー側では、showmount により、NFS クライアントの利用状況を把握する事ができます。「-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。

```
# 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 エクスポートディレクトリのアンマウント

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。CD-ROM な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/export を作成して、エクスポートを行います。

1. ディレクトリ/work を作成します。
2. 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。
3. nfs サービスを起動します。
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、showmount 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、/work 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 を指定する事も可能である。
async	エクスポートディレクトリ内のファイルへのアクセスを非同期でおこなう。
sync	クライアント側の変更は直ちにサーバー側の実ファイルに反映される（デフォルト）
wdelay	複数の書き込み処理を一度に行わない（デフォルト）
no_wdelay	複数の書き込み処理を一度に行う。ただし、rw 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である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/export ディレクトリに対する設定を/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」オプションを渡して実行します。

```
# exportfs -r
# exportfs -v
/work           10.33.0.0/255.255.0.0(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のアクセスを許可し、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. /tmp 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します。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
```

上の図の例を見ればわかる様に、UID を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、passwd ファイルや 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=65536)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 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。

```
# exportfs -rv
exporting 10.20.0.0/255.255.0.0:/tmp
exporting 10.20.0.0/255.255.0.0:/work
# mount /media/nfs -o remount
# 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 マウントを解除します。
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 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、隣の方の/home ディレクトリを、/media/nfs ディレクトリにマウントします。
5. お互いのホストの 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。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」を付加して、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. NFS でマウントされたファイルシステム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。本章の FTP サーバーに関連する、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しょう。

7.1.1 パッケージの入手

OSS は一般的に tarball の形式で、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 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。

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

```
# ftp   ftp.riken.jp
Connected to riksun.riken.go.jp.
220 ::ffff:134.160.38.1 FTP server ready
500 AUTH not understood
500 AUTH not understood
KERBEROS_V4 rejected as an authentication type
Name (ftp.riken.jp:root): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230-**********************************************************
 ftp.riken.jp is an unsupported ftp/http/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
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」。パスワード替わりに「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。

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 (134,160,38,1,192,142).
150 Opening ASCII mode data connection for file list
drwxr-xr-x   5 0        0            4096 Aug  6  2008 11
               [中略]
drwxr-xr-x  17  archive  archive       4096 Jul 21  2008 net
drwxr-xr-x   8  archive  archive       4096 Jul 25  2008 pc
drwxr-xr-x   4  archive  archive       4096 Nov 29 10:27 pub
drwxrwxr-x  20 archive  archive        4096 May  8 23:10 tex-archive
-rw-r--r--    1  root      root          642 Apr 30 05:46 welcome.msg
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 (134,160,38,1,192,155).
150 Opening ASCII mode data connection for file list
drwxr-xr-x   4 archive  archive      8192 May  8 17:36 OpenSSH
drwxr-xr-x   2 archive  archive     12288 May  8 17:36 OpenSSL
drwxr-xr-x   6 archive  archive      4096 May  8 17:28 ProFTPD
               [中略]
226 Transfer complete
ftp> cd   ProFTPD
250 CWD command successful
ftp> ls
227 Entering Passive Mode (134,160,38,1,192,168).
150 Opening ASCII mode data connection for file list
-rw-r--r--  1 archive  archive       11 May   8 17:10 MIRMON.PROBE
-rw-r--r--  1 archive  archive      451 Jul   1  2005 README.MIRRORS
drwxr-xr-x  3 archive  archive     4096 Jul   1  2005 contrib
drwxr-xr-x  3 archive  archive     4096 Jul   1  2005 devel
drwxr-xr-x  4 archive  archive     4096 Nov  19  2002 distrib
drwxr-xr-x  4 archive  archive     4096 Jul   1  2005 historic
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 cd します。

```
ftp> cd   distrib
250 CWD command successful
ftp> ls
```

```
227 Entering Passive Mode (134,160,38,1,192,183).
150 Opening ASCII mode data connection for file list
drwxr-xr-x   4 archive  archive      4096 Jan 30  2003 packages
drwxr-xr-x   2 archive  archive      4096 Feb  5  19:19 source
226 Transfer complete
ftp> cd   source
250 CWD command successful
```

あらかじめ調べておいた最新版の 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。(この資料作成時点での最新版は proftpd-1.3.5rc4.tar.gz です)

```
ftp> ls
227 Entering Passive Mode (134,160,38,1,192,196).
150 Opening ASCII mode data connection for file list
-rw-r--r-- 1 archive  archive   1386956 Nov 27  2006 proftpd-1.3.0a.tar.bz2
               [中略]
-rw-r--r-- 1 archive  archive   7432702 Jun 14  2013 proftpd-1.3.5rc3.tar.gz
-rw-r--r-- 1 archive  archive       197 Jun 14  2013 proftpd-1.3.5rc3.tar.gz.asc
-rw-r--r-- 1 archive  archive        58 Jun 14  2013 proftpd-1.3.5rc3.tar.gz.md5
-rw-r--r-- 1 archive  archive   7580690 Jan 28  2014 proftpd-1.3.5rc4.tar.gz
-rw-r--r-- 1 archive  archive       197 Jan 28  2014 proftpd-1.3.5rc4.tar.gz.asc
-rw-r--r-- 1 archive  archive        58 Jan 28  2014 proftpd-1.3.5rc4.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が完了したら、quit コマンドで終了します。

```
ftp> get   proftpd-1.3.5rc4.tar.gz
200 PORT command successful
150 Opening ASCII mode data connection for proftpd-1.3.5rc4.tar.gz (7580690 bytes)
226 Transfer complete
7580690 bytes received in 0.792 secs (9565.64 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事を確認します。

```
$ ls   -l   proftpd-1.3.5rc4.tar.gz
-rw-rw-r-- 1 student student 7580690  7 月 30 09:38 proftpd-1.3.5rc4.tar.gz
```

より正確に確認するには、.md5 ファイルもダウンロードし、その値が md5sum コマンドの実行結果と一致することを確認します。

```
ftp> get   proftpd-1.3.5rc4.tar.gz.md5
local: proftpd-1.3.5rc4.tar.gz.md5 remote: proftpd-1.3.5rc4.tar.gz.md5
227 Entering Passive Mode (134,160,38,1,140,3).
150 Opening BINARY mode data connection for proftpd-1.3.5rc4.tar.gz.md5 (58 bytes)
226 Transfer complete
58 bytes received in 0.0121 secs (4.77 Kbytes/sec)
```

```
ftp> quit
221 Goodbye.
$ cat  proftpd-1.3.5rc4.tar.gz.md5
8e9241bda2d31dfb3ed18e5108431b57  proftpd-1.3.5rc4.tar.gz
$ md5sum  proftpd-1.3.5rc4.tar.gz
8e9241bda2d31dfb3ed18e5108431b57  proftpd-1.3.5rc4.tar.gz
```

［練習］

どのサイトからダウンロードしてもバージョンさえ同じならパッケージファイルの内容に違いはありません。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チェック

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-1.3.5rc4.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-1.3.5rc4/
# ls
COPYING         README.Solaris2.5x   config.status  locale
CREDITS         README.Unixware      config.sub     ltmain.sh
ChangeLog       README.capabilities  configure      m4
INSTALL         README.classes       configure.in   module-libs.txt
          ［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 が不足するようであれば、yum を使ってインストールしておいてください。

```
#  ./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
```

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、出来上がった 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 の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-1.3.5rc4/lib' に入ります
gcc -DHAVE_CONFIG_H  -DLINUX  -I.. -I../include  -O2 -Wall -c pr_fnmatch.c
In file included from pr_fnmatch.c:256:0:
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  -O2 -Wall -c sstrncpy.c
gcc -DHAVE_CONFIG_H  -DLINUX  -I.. -I../include  -O2 -Wall -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 という引数を渡して実行します。

```
# make  install
cd lib/ && make lib
make[1]: ディレクトリ `/home/student/proftpd-1.3.5rc4/lib' に入ります
make[1]: `lib' に対して行うべき事はありません.
make[1]: ディレクトリ `/home/student/proftpd-1.3.5rc4/lib' から出ます
cd src/ && make src
make[1]: ディレクトリ `/home/student/proftpd-1.3.5rc4/src' に入ります
make[1]: `src' に対して行うべき事はありません.
          [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が、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 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の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くくら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アウト扱いとなり、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 Installatio)
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。(no)
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］

今回の ProFTPD の設定ファイル/usr/local/proftpd/etc/proftpd.conf を開きます。ProFTPD のプロセスのユーザー権限とグループ権限の設定値を確認します。グループ権限にシステムに存在しないグループ名「nogroup」が設定されていたら、「nobody」に変更しておいてください。

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
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:2049            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:59781           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
23037/proftpd: (acc
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
1145/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
1259/master
                  ［省略］
```

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 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
STOR	クライアントからサーバーへのファイル転送
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の取得
READ	ファイルの読み込み
WRITE	ファイルやディレクトリの作成や削除
ALL	すべての FTP コマンド

<Limit コマンド>～</Limit>中では、Allow、AllowAll、AllowUser、AllowGroup、Deny、DenyAll、DenyUser、DenyGroup、HideUser、HideGroup 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。

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、以下のようにします。

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

［練習］

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.**.0/16（**は教室のネットワークに対応します）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 その他のセキュリティ設定（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、設定を変更すると便利です。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 その他のセキュリティ設定（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・パスワード情報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］

上記のように 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・運用する上で、サーバーを適切に構築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どう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、起動後の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どの程度行っているのか、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項目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 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)
  Active: active (running) since 木 2015-07-30 11:09:39 JST; 4s ago
 Main PID: 23516 (rsyslogd)
  CGroup: /system.slice/rsyslog.service
       mq23516 /usr/sbin/rsyslogd -n

 7月 30 11:09:39 h006.s142.la.net systemd[1]: Started System Logging Service.
```

システム管理の基本となるサービスですので、停止させてしまう事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
セレクタ（情報の選択） アクション（出力先）
```

8.1.2 ログのセレクタ

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。これをセレクタと呼び、ドット（.)で接続して表記します。ファシリティ、プライオリティを複数指定する場合はカンマ（,）で区切り、セレクタ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　　ファシリティの uucp または news　で、プライオリティ crit

*.info;mail.none;　　ファシリティに関わらず、プライオリティ info または mail.none

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

プライオリティレベルは、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。もっと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します。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含んだことになるのです。

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）あらゆるチャネルから通知
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）

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・プライオリティごとに、どんな処理を行うかを決める事ができます。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	指定されたデバイスへ出力	/dev/console
| コマンド	コマンドをパイプラインで接続	| /bin/cat -n
ユーザー名	指定されたユーザーにメールで通知	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。個々での例では、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。
/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 よりもプライオリティが高いログを、/var/log/messages に記録することを意味しています。

［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、syslog 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 コマンドを実行し、ユーザー logtest を作成します。この時、作成したログ情報が指定した画面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、定めた日時のジョブ実行や、定期的は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 というデーモンで構成されています。cron の設定は crontab コマンドを利用することで行います。

例えば、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、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 の使用方法を使って作業を行います。

```
$ crontab -e
  *  *  *  *  *  echo ”cron test” > /dev/tty1
```

これは、毎分「echo "cron test" > /dev/tty1」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、5

0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
* * * * * date >> /dev/pts/1
```

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 が定期的に実行するのに対して、at は1度のみ実行します。at に登録したジョブは、atd デーモンによって制御されます。

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]で入力終了を教えます。

時間指定の例

```
at  12:00              12:00 ちょうどに実行
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       11/15 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。

予約してあるジョブを取り消すには、「atrm ジョブ No.」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>           ← ^D を入力すると<EOT>と表示されます。
job 1 at Fri Jul 31 03:20:00 2015
$ at 4:20 +1 days
at> date
at> <EOT>
job 2 at Fri Jul 31 04:20:00 2015
$ atq
1       Fri Jul 31 03:20:00 2015 a student
2       Fri Jul 31 04:20:00 2015 a student
$ atrm 2
$ at -l
1       Fri Jul 31 03:20:00 2015 a student
```

8.3 NTP

8.3.1 NTP サーバー

サーバーは、ログの記録やジョブの定時実行のため、正しい時刻を把握しておくことが重要といえます。ログの記録されている時刻が正しくなければ、ログに対応するトラブルやイベントがいつ行われたのかがわからなくなりますし、cron や at で実行する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。

したがって、通常、サーバーは NTP（Network Time Protocol）と呼ばれるプロトコルを利用して、正確な時刻を保持するようにします。

NTP を利用すると、ネットワーク上にある NTP サーバーの時計と時刻を同期することができます。NTP サーバーは、原子時計を内部に搭載した GPS（Global Positioning System）衛星などから信号を受信することで、世界標準時に同期した高い時刻精度を維持しています。

NTP サーバーにアクセスすると、NTP サーバーから時刻情報を取得することができ、通信にかかった時間を考慮して、時刻情報を設定します。こうすることで、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。

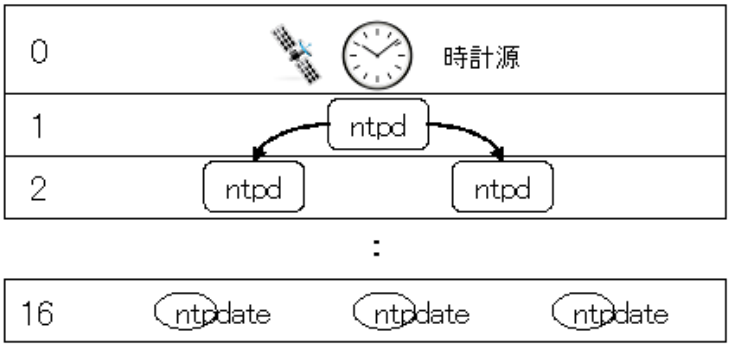
NTP は階層構造をとる事ができ、原子時計や GPS を起点（Stratum 0)とし、そこに直結しているものを Stratum 1、さらにそのサーバーを参照しているものを Stratum 2 とし最大16層まであります。Stratum 16 はクライアントのみで他のサーバーにはなれません。

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 サーバーを参照するときは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、このサイトを訪れるとよいでしょう。

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
2015 年  7 月 30 日 木曜日 14:09:58 JST
# ntpdate 0.jp.pool.ntp.org 1.jp.pool.ntp.org
30 Jul 14:10:19 ntpdate[25528]: adjust time server 157.7.236.66 offset 0.000283
sec
```

8.3.3 ntpd

NTP サーバーとして配信する事も可能です。その場合は /etc/ntp.conf ファイルのサーバーに、先の上位 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
```

ntpq コマンドにより、同期の状況を表示する事ができます。

```
# systemctl restart ntpd
# ntpq -p
   remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*y.ns.gin.ntt.ne 129.250.36.  2 u   -   64    1   10.355   -0.787   0.061
 v157-7-235-92.z 10.84.87.14  2 u   1   64    1    5.162   -0.962   0.002
 7c2956c8.i-revo 133.243.238  2 u   -   64    1    8.556    0.430   0.002
 hndpdccp01.corp .INIT.      16 u   -   64    0    0.000    0.000   0.002
```

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、m:マルチ、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. ユーザー root に移行します。

3. date コマンドを実行して、システムクロックを確認します。

4. 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。

5. 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、毎日午前2時10分に ntpdate コマンドを実行するようにします。

6. crontab に次の設定を追加します。
   10　2　*　*　*　/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、syslog に出力するようになります。

8.4 オンラインマニュアル

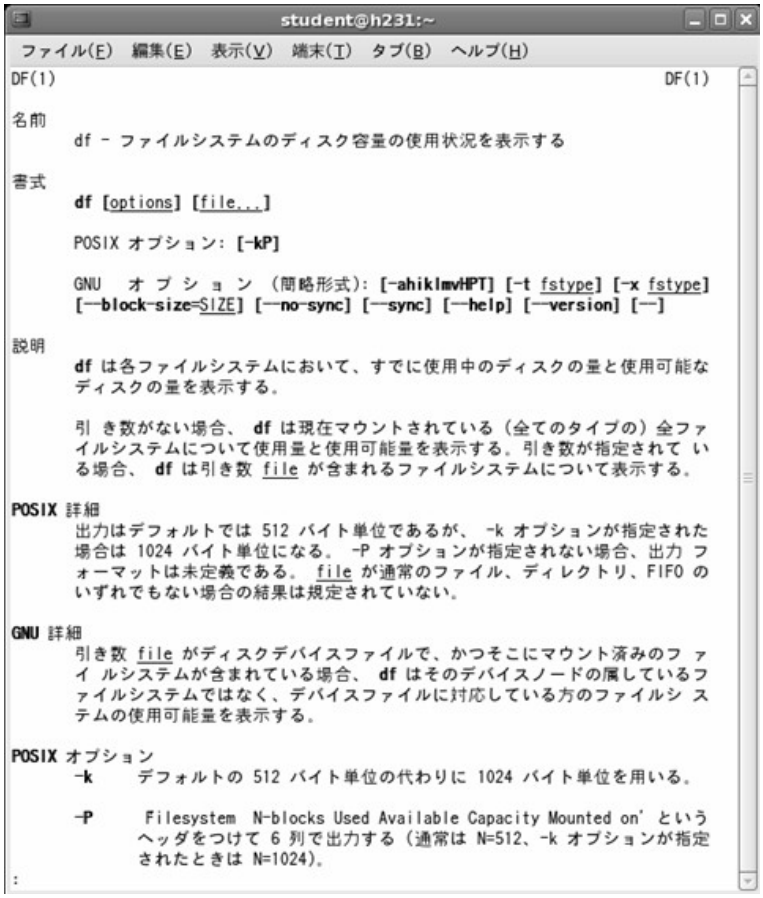
8.4.1 オンラインマニュアルとは

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



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 2014 年 6 月版、日本語翻訳版は 2012 年 4 月版となっています。

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属するかが決まっています。

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、sa、st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章に所属しており、passwd コマンドは第1章に所属しています。

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	名称	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	解説	コマンドやファイルの解説
OPTIONS	オプション	コマンドオプションの解説
FILES	関連ファイル	関連するファイル
SEE ALSO	関連事項	関連する他のマニュアル
BUGS	バグ	既知の問題点

マニュアルのデータは /usr/shar/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にあり、日本語の場合は /usr/share/man/ja/man 章番号となります。

8.4.4 オンラインマニュアルの関連コマンド

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
```

の2つのコマンド「whatis」「apropos」はともに、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「makewhatis」コマンドによって更新できます。「makewhatis」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。

8.4.5 info コマンド

GNU のツールについては、info コマンドでもマニュアルが提供されています。引数にコマンドを指定し起動します。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: Directo¥
ry 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.
```

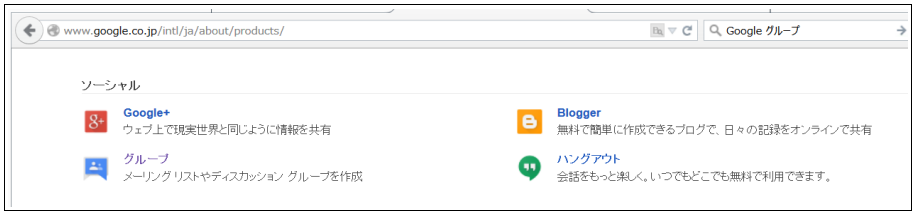
Info は Emacs と呼ばれる高機能エディタを利用しており、主な操作は以下のようになります。

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
^S	順方向へ検索（インクリメンタルサーチ）
^R	逆方向へ検索
q	修了

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。

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事ができます。
グーグルサービス一覧（http://www.google.co.jp/intl/ja/about/products/）から、ソーシャルのグループをクリック



「すべてのグループから」、カテゴリを指定することで絞り込みができます。

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ものは、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やツールが用意されています。これらのツール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 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:         783072      184292      411808        5488      186972      447088
Swap:       1023996           0     1023996
```

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事ができます。-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/dm-1                               partition       1023996 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、ファイルをスワップ領域として割り当てることも可能です。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。

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）
```

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で、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)
  r は実行中(running)、b(blocked)は入出力待ち のプロセス数を表します。

- memory　メモリの利用状況(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データが転送された回数、so（スワップアウト）はメモリからスワップ領域への回数

- 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(storen)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        1K-ブロック   使用  使用可 使用% マウント位置
/dev/mapper/centos-root     1020588   65680  954908    7% /
devtmpfs                     381804       0  381804    0% /dev
tmpfs                        391536       0  391536    0% /dev/shm
tmpfs                        391536    5484  386052    2% /run
tmpfs                        391536       0  391536    0% /sys/fs/cgroup
/dev/mapper/centos-usr      5109760 3528524 1581236   70% /usr
/dev/sda1                    127660   89276   38384   70% /boot
                （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新規にファイルを作成することができなくなるので、ディスク使用率だけでなく、i ノードの使用状況にも注意を払う必要があります。

```
$ df -i
ファイルシス        I ノード I 使用  I 残り I 使用% マウント位置
```

```
/dev/mapper/centos-root 1024000   2765 1021235     1% /
devtmpfs                  95451    358   95093     1% /dev
tmpfs                     97884      1   97883     1% /dev/shm
tmpfs                     97884    438   97446     1% /run
tmpfs                     97884     13   97871     1% /sys/fs/cgroup
/dev/mapper/centos-usr  5120000 122468 4997532     3% /usr
/dev/sda1                131072    329  130743     1% /boot
                （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/
40K     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
20K     public_html/svc1.lis
16K     public_html/svc2.lis
40K     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。

9.2 冗長性のあるシステム構成の構築

コンピューターは電子機器です。どんなに良質な部品を使っていても故障する可能性はあります。故障によって、サービスが提供できない、データが失われる、といった重大な損失につながりかねません。

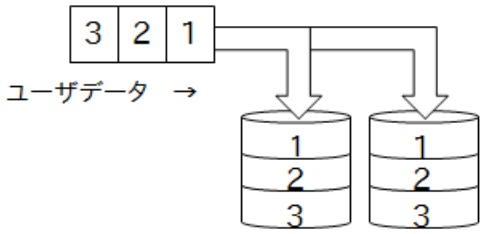
システムの冗長化を行うことによって、一部に障害が発生した場合にも継続してシステムを動作させることが可能になります。

9.2.1 RAID

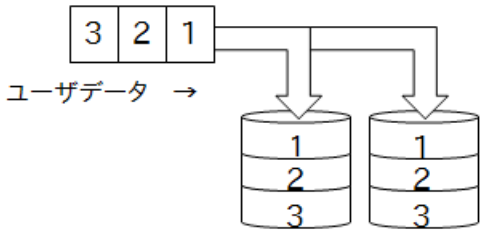
複数のハードディスクを組み合わせて 1 台のハードディスクのように扱う技術を RAID（Redundant Arrays of Independent/Inexpensive Disks）といいます。RAID によってディスクの冗長化や、大容量化、高速化を行うことが可能です。

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。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が、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 カーネルはバージョン 2.4 で、RAID 0、RAID 1、RAID 5 を、バージョン 2.6 で上記3種に加え、RAID 6 の計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、複数のディスクへの同時並行的な処理が可能になり、高速に読み書きが行えます。また、複数のディスクを1台のディスクのように扱えるため、大容量のディスクとして扱うことができます。しかし、肝心の冗長性は確保されておらず、どれか1台のディスクが故障することで、ディスク全体が読み出し不能となる欠点があります。

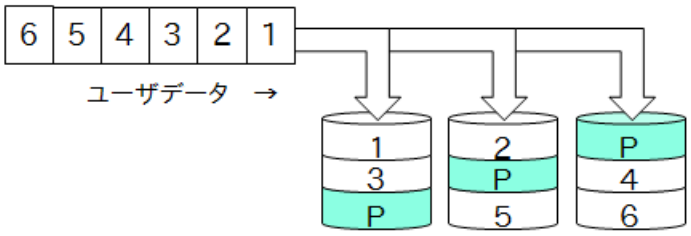


RAID 1（ミラーリング）
同一のデータを2台以上の複数のディスクに書き込みます。1台のディスクが故障しても、他のディスクが処理を続行できるため、耐障害性に優れています。しかし、同じデータを台数分のディスクに対して書き込むため、ディスクの使用効率は1/台数分になってしまうという欠点があります。



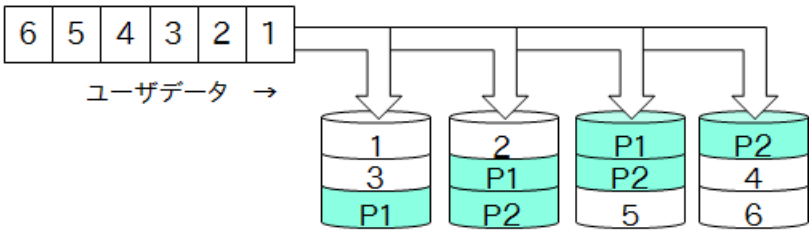
RAID 5
データと、そのデータから生成されたパリティ情報を3台以上の複数ディスクに分散して書き込みます。万一ディスクの1台が故障しても、残ったデータとパリティ情報から、失われたデータの復元ができるた

め、システムの復旧が行えます。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ディスクが使用できますし、同時並行的な処理も行っていることから、高速な読み書きも可能になります。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。



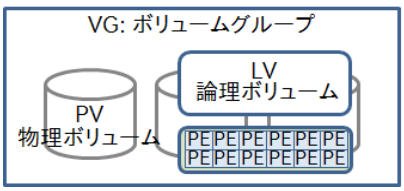
RAID 6

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

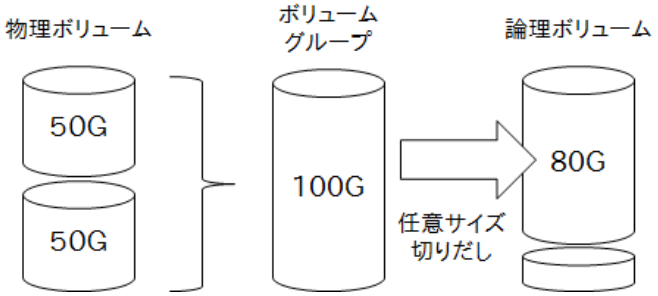
LVM（Logical Volume Manager）は、固定的なパーティションの制限超え、柔軟なディスク管理を実現する技術です。複数のハードディスクを「ボリュームグループ」としてまとめ、そこからパーティションに相当する「論理ボリューム」を切り出して使用します。



- 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性が出てきます。

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事も可能です。

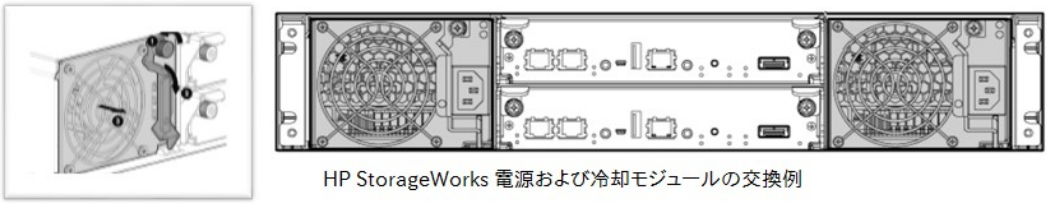


LVM によって柔軟なディスク管理が可能となりますが、冗長化を確保することはできません。前述の RAID と LVM を組み合わせることによって、冗長性を確保しつつ、柔軟なディスク管理を行うことが可能になります。

9.2.3 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、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

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 バックアップ対象となるディレクトリ

ハードディスクの大容量か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

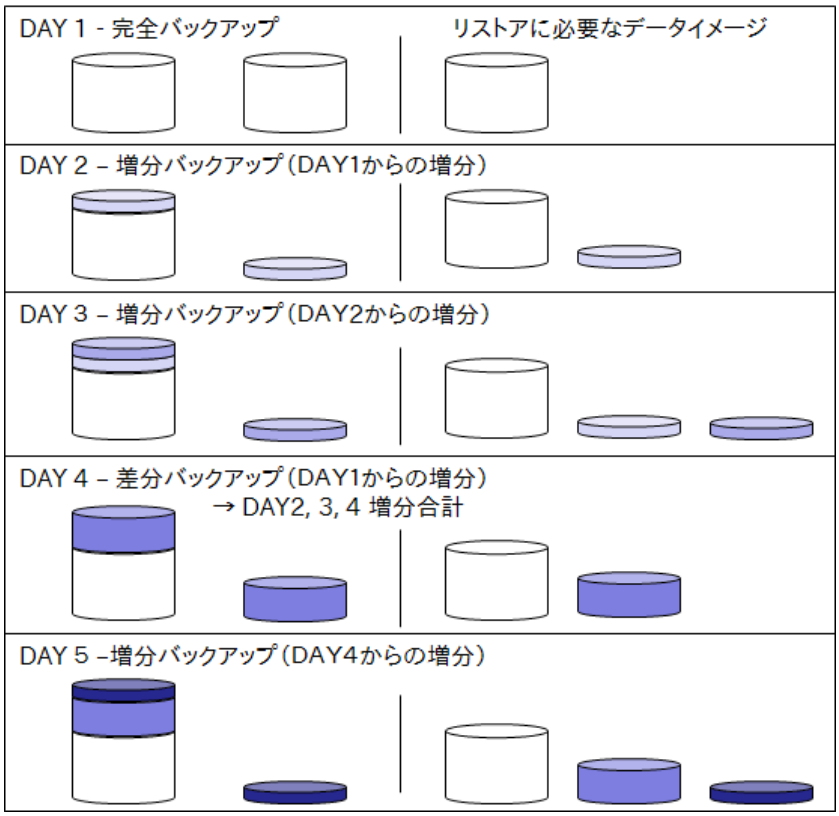
バックアップには、次に説明するように、いくつかの種類があります。これらを使い分けることで、バックアップのサイズやバックアップ処理に要する時間、リストア（バックアップからの復旧）作業にかかる手間のバランスを取ることができます。

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。



容量が大きくな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事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
-l 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。たとえば日曜日に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。
-f 出力ファイル	ダンプしたデータを出力する先を指定。
-L 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。
-M 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。
-J	ダンプの明細を取得しない（明細は /var/lib/xfsdump/inventory 下へ格納）

実行例

```
# xfsdump -l 0 -L "boot backup" -M "disk" -f /var/boot.bkup /boot
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) : 83993280 bytes
xfsdump: dump complete: 0 seconds elapsed
xfsdump: Dump Summary:
xfsdump:   stream 0 /var/boot.bkup OK (success)
xfsdump: Dump Status: SUCCESS
```

最後が SUCCESS（成功）で終わっている事を確認しましょう。

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回復する例

```
# 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:
 -> ls
       139 initramfs-3.10.0-229.el7.x86_64.img
                           :
       136 symvers-3.10.0-229.el7.x86_64.gz
                           :

 -> add symvers-3.10.0-229.el7.x86_64.gz
 -> 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/boot0.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、任意のファイルを回復する。

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。守るべきものが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。

第一にハードウエアを守ること。もし、サーバー機器が盗難にあったらネットワーク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。極論すればサーバーやネットワーク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、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、出来う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

ハードウエ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。データセンタは外界と遮断されており、教育・訓練を受けた千人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を設けるなど、データセンタの需要が伸びています。

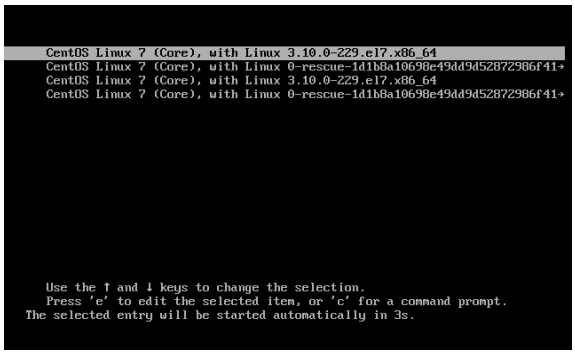
10.2.2 コンソールの横取り

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事が可能です。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] を入れると、ブートパラメータを修正する事ができます。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事ができます。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事が可能です。

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。



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。

[^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。

```
[  OK  ] Reached target Initrd Default Target.
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 menu のパスワード保護

上記の不正操作を回避するために、Boot menu に認証機能を追加します。

/etc/grub.d/10_linux スクリプトを別名で保管したのち、末尾に以下を追加します。

```
##### boot menu password
cat <<EOD
set superusers=”maint”          ←ユーザー名
password  maint  linux          ←ユーザー名に対するパスワード
EOD
```

GRUB2では、/etc/grub.d/ 以外にも、初期値を設定する /etc/default/grub があります。これらを修正した場合は、整合を確保するため grurb2-mkconfig で新しい起動用スクリプトを生成します。

/boot/gurb2/grub.cfg を別名で保管したのち、以下のコマンドを実行し再起動します。

```
#  grub2-mkconfig -o /boot/gurb2/grub.cfg
```

[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. /etc/grub.d/10_linux を修正します。(上記内容)
5. OSを再起動し Boot menu を操作します。パスワード入力が行われる事を確認します。
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もっと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あったりします。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、利害目的が比較的はっきりしていますから的を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事があります。

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

ソフトウェアのバグ・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.jpcert.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 として yum update コマンドを実行して、パッケージを最新の物にします。

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、稼動させない事が基本です。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はありますか。

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事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、広く公開する事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ごてに回る事があります。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。xinetd 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こない限り、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 xinetd 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起動対象から外します。
5. システムを再起動します。
6. もう一度稼働中のサービスを確認します。

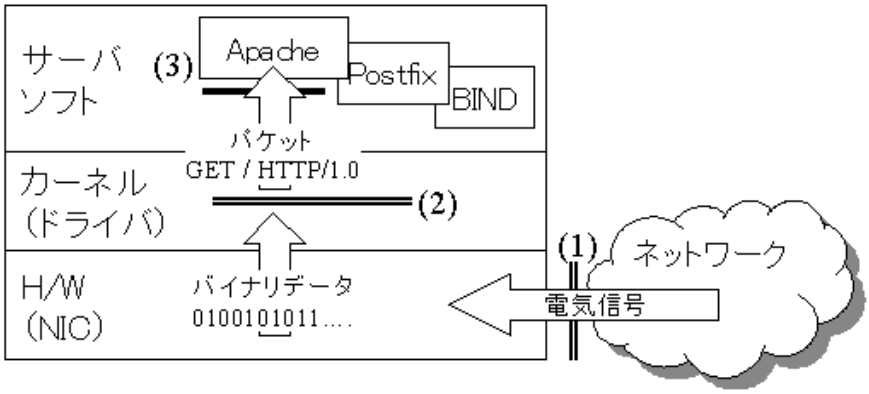
開いているポートがどのサービスによるものなのか判断がつかない場合があります。この場合、「lsof -i:ポート番号」コマンドを実行して、動作中のプログラムの PID を調べ、場合によっては kill します。

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かうデータは「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ーは、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ー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制限する。
2. ネットワークドライバによって制限する。
3. サーバーソフトで制限を行う。

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、hosts.deny ファイルに記述しました。Apache HTTP サーバーも、Order、Allow、Deny ディレクティブで、BIND は各種 allow オプションによって、送信元の情報を基にアクセス制限を行います。

これ以外に、サーバーコンピューターに届く以前、つまりNICに入ってくる段階（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になります。

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、その為にはどの様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、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。

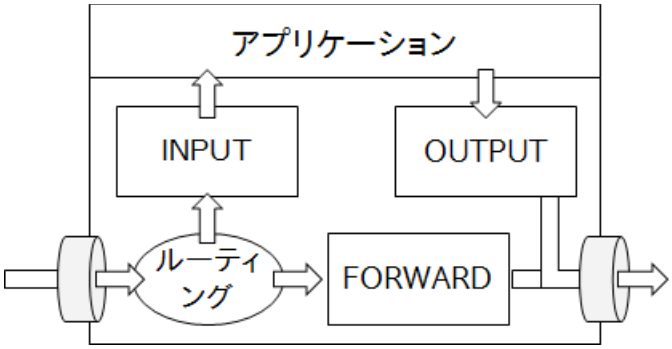
代表的なサービスとプロトコル、ポート番号

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
Netbios	TCP, UDP	137-139, 445
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「チェイン」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 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 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
    # 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。

```
# 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 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。
REJECT	当該パケットを遮断しエラーを通知する。
DROP	当該パケットを廃棄（エラーは通知しない）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。

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、「pollicy 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インにおいて、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。

チェイン	ターゲット
INPUT	DROP
FORWARD	DROP
OUTPUT	ACCEPT

チェイ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」はチェイン、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になります。

［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. 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。
   続いて、dig コマンドや ping コマンドで www.google.com について問い合わせます。
4. ping コマンドでローカルホストに対して、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たら、INPUT チェインのポリシーを ACCEPT に戻します。

10.4.4 iptables コマンドによる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	宛先アドレス	送信元ポート	宛先ポート
TCP	10.20.142.6	Any	80
TCP	10.20.142.6	Any	53
UDP	10.20.142.6	Any	53
TCP	10.20.142.6	Any	25
TCP	10.20.142.6	Any	110
TCP	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	宛先アドレス	送信元ポート	宛先ポート
TCP	10.20.142.6	80	Any
TCP	10.20.142.6	53	Any
UDP	10.20.142.6	53	Any
TCP	10.20.142.6	25	Any
TCP	10.20.142.6	110	Any
TCP	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	宛先アドレス	送信元ポート	宛先ポート
Any	127.0.0.1	Any	Any
ICMP	Any	ー	ー

この節では、上記ルールを 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。

例えば、送信元アドレスが www.example.net で宛先ポート番号が 80 番の TCP パケットを指定するためには以下のようなオプションを用いることになります。

```
      -p tcp  -s www.example.net  --dport 80
```

該当するパケットの行き先であるターゲットを指定するには、「-j」オプションを用います。

以上を踏まえて、「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
   ```

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、各チェイ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
```

各チェイ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、全チェインを対象にする場合はチェイン名を省略します。

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

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