サーバー基礎コース

はじめに

Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロです。Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロです。Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロです。Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロです。
Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロで
Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロです。Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロです。Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロです。Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロです。
Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロで
Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロです。Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロです。Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロです。Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロです。
Word 2000 は、画期的な日本語入力・編集環境を実現した日本語ワープロで

本講座が皆様のお役にたてれば幸いに存じます。

Contents

1章　Unix システム

1 UNIX とは

1.1 UNIX の歴史

UNIX と呼ばれる OS は 1968 年に米 AT&T ベル研究所にて「初のマルチタスク・マルチユーザ OS」として開発されたものから始まりました。

初のマルチタスク、マルチユーザ OS

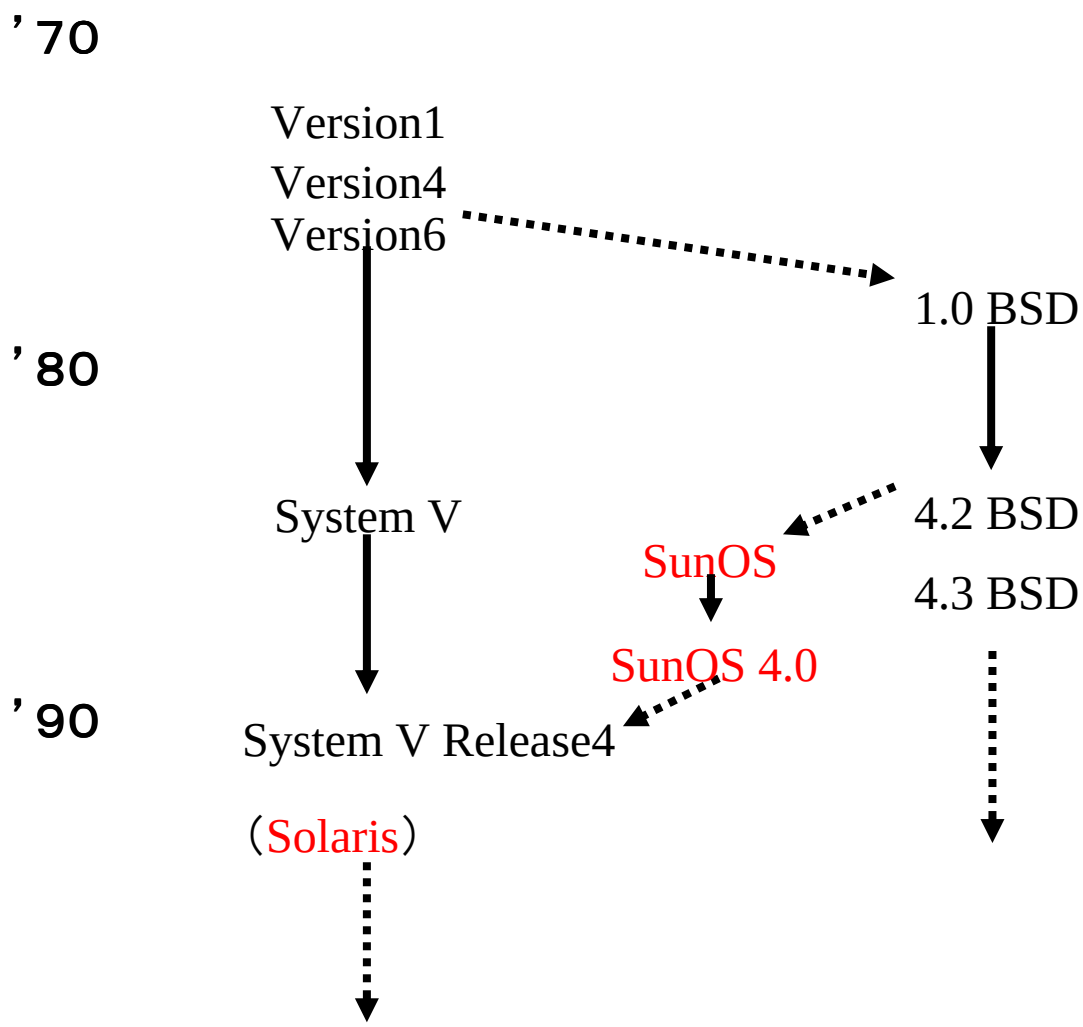
当初は、DEC の PDP-7 上で動作していましたが、1973 年アセンブラから C に書換えられ移植性が高まり、他のプラットフォームでも動作する OS となりました。

最初は社内利用を目的として開発されましたが、その後ソースコードと開発ツールをパッケージ化し教育機関に無料で配布されました。大学等を中心に広がるとともに、微妙に異なる OS に派生し、進化していきました。有名なものとしては、カリフォルニア大学バークレイ校の BSD があります。

一般的に国内企業系では、Solaris・各種 BSD・Linux を含めて UNIX として語られています。通常われわれが UNIX と呼んでいる OS は UNIX および UNIX 互換 OS ということになります。（以降、本書では「UNIX および UNIX 互換 OS」のことを UNIX と称します。）

基本的には、Solaris(SystemV)系統と BSD 系統に区分されます。

- 対外部向けに各種サービス提供を検討する場合、Solaris を候補に挙げる事例が多い
- 対内部向け各種サービスの為に検討する場合は、BSD 系統を選択する場合が多い


'70
Version1
Version4
Version6
1.0 BSD
'80
System V
4.2 BSD
SunOS
4.3 BSD
SunOS 4.0
'90
System V Release4
(Solaris)

1.2 UNIX の種類

通常われわれが UNIX と呼んでいる OS は UNIX および UNIX 互換 OS ということになります。

UNIX

1968 年にアメリカ AT&T 社のベル研究所で開発された OS です。C 言語というハードウェアに依存しない移植性の高い言語で記述され、またソースコードが比較的コンパクトであったことから、多くのプラットフォームに移植されました。また学術機関やコンピュータメーカーの手によって独自の拡張が施された多くの派生 OS が開発され、現在では UNIX 風のシステム体系を持った OS を総称的に UNIX と呼んでいます。代表的なものだけでも、Sun Microsystems 社の Solaris と SunOS、Hewlett Packard 社の HP-UX、IBM 社の AIX、SGI 社(旧 Silicon Graphics 社)の IRIX、Santa Cruz Operations 社の UnixWare、カリフォルニア大学バークリー校(UCB)の BSD(と FreeBSD などの派生 OS)、Linus Torvalds 氏の Linux などがあります。

商標としての UNIX は業界団体 The Open Group が所有しており、SPEC1170 と呼ばれる技術仕様を満たした OS のみが、正式に「UNIX」を名乗れることになっています。また、各 UNIX クローン OS 間の互換性を確保するため、国際標準化機構(ISO)によって最低限備えるべき技術仕様 POSIX がまとめられています。UNIX は一般に、①完全なマルチタスク機能を搭載し、②ネットワーク機能や安定性に優れ、③セキュリティ強度が高いことで知られています。また、1 台のコンピュータを複数の人間で同時に使用することを前提に設計された④マルチユーザ OS であり、ネットワークを通じて端末機から作業をすることができます。UNIX 及びその互換 OS は学術機関や企業の研究所などを中心に広く普及しており、データベースなどの大規模なアプリケーションソフトが豊富なことから、企業の基幹業務用のサーバとしても多く採用されています。

BSD

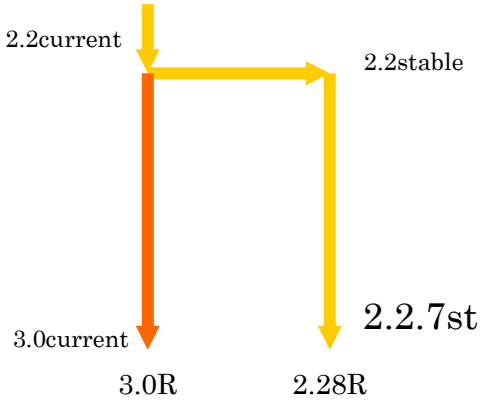
カリフォルニア大学バークリー校で開発された UNIX 互換 OS です。多くの派生 OS がありインターネットサーバ用 OS や研究用の OS として広く普及しています。386BSD として PC(PC/AT 互換機)にも移植され、その後 PC 用の UNIX 互換 OS として 386BSD の後継である FreeBSD や NetBSD が広く普及しています。

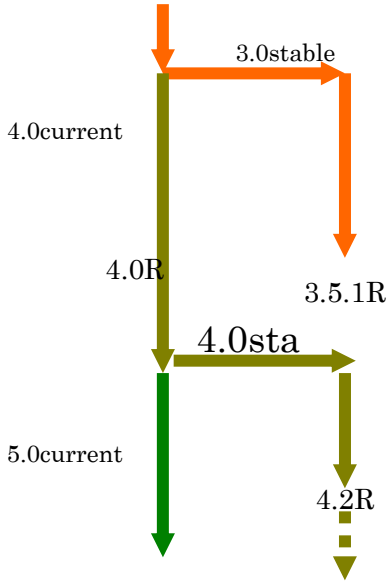
FreeBSD

1993 年に Nate Williams 氏・Rod Grimes 氏・Jordan K. Hubbard 氏の 3 人によって開発が始まった PC/AT 互換機用の UNIX 互換 OS です。現在では PC/AT 互換機以外のプラットフォームにも移植されています。FreeBSD はフリーソフトウェアとしてソースコードと共に無償で公開されており、全世界のボランティア・プログラマの手によって開発が進められています。

FreeBSD の最初のバージョンはカリフォルニア大学バークリー校(UCB)の 4.3BSD から派生した Net/2 という OS をベースに開発されましたが、Net/2 の一部に UNIX System Laboratories(USL)が著作権を持つコードが含まれていたため、4.4BSD-Lite をベースとしたものに移行しました。FreeBSD は安価な PC/AT 互換機上で動作し、他の UNIX 互換 OS と同様に優れたセキュリティや高い安定性を備えているため、インターネットサービスプロバイダや学術機関などを中心に導入が進んでいます。FreeBSD は Linux と並んでインターネットサーバとして人気のある OS です。FreeBSD は BSD ライセンスというライセンス体系に基づいて公開されているため、GNU プロジェクトの GPL に基づいて公開されている Linux などよりも利用に関する自由度が高くなっています。

FreeBSD は世界中の開発者によって毎日更新されています。新しい機能の追加などの変更はまず current バージョンに反映されます。多くのユーザに安定した環境を提供するのが stable バージョンです。





NetBSD

カリフォルニア大学バークリー校(UCB)の 4.4BSD-Lite をもとに、Chris G. Demetriou 氏らが中心となって開発が始まった UNIX 互換 OS です。同じく 4.4BSD-Lite をもとに開発された FreeBSD が PC/AT 互換機にハードウェアを絞り込んでいるのとは対照的に、移植性を高めできるだけ多くのハードウェアをサポートすることを目指しています。このためサポートしているハードウェアの数が非常に多いのが特徴です。より先進的な機能を重視し積極的に新しい機能を取り込んでいます。フリーソフトウェアとしてソースコードと共に無償で公開されており、全世界のボランティア・プログラマの手によって開発が進められています。他の UNIX 互換 OS と同様、優れたセキュリティや高い安定性を備えているため、学術機関などを中心に導入が進んでいます。

Solaris

SunSoft 社(Sun Microsystems 社の子会社)が開発・販売している UNIX 系 OS です。Sun Microsystems 社製のコンピュータで動作するほか、PC/AT 互換機で動作するバージョンもあります。同社は Solaris 以前に SunOS という BSD 系 OS を開発していましたが、SunOS 5.x から System V 系 OS に変更しました。BSD 系最後のリリースにあたる SunOS 4.1.4 には Solaris 1.1.2 という名称が与えられ、System V 系の SunOS 5.6 には Solaris 2.6 という名称が与えられました。最近では「SunOS」は Solaris のカーネル部分を指す言葉として用いるようになってきています。

POSIX

IEEE によって定められた UNIX ベースの OS が備えるべき最低限の仕様のことです。各社の UNIX 互換 OS にはそれぞれ独自の拡張や仕様の変更が施され互換性が失われてしまったため、各 OS 間で最低限の互換性を確保するために定義されました。アプリケーションソフトが OS の提供する機能を呼び出すための方法(システムインターフェース)などを定義しています。アメリカ規格協会(ANSI)や国際標準化機構(ISO)でも標準として採用され、アメリカ政府機関に納入する UNIX システムが守るべき必須条件となっています。

HP-UX

Hewlett Packard 社の UNIX 互換 OS です。同社の RISC 方式のマイクロプロセッサ「PA-RISC」シリーズを搭載したコンピュータで動作します。Intel 社と共同開発中の 64 ビットプロセッサ「Merced」は PA-RISC シリーズの上位互換となっており、HP-UX も移植される見通しです。

AIX

IBM 社の開発した UNIX 互換 OS です。同社の RS/6000 シリーズなどに搭載されています。AT&T 社の System V をベースにしています。同社のメインフレームシステムや PC 用 OS の OS/2 との連携に優れています。

Xopen

UNIX を基盤としたシステムの標準化を図るために 1984 年に設立された業界団体です。1987 年にイギリスに本社を置く会社組織になりました。UNIX の標準化をめぐって対立する 2 つの勢力、AT&T 社などを中心とする UNIX International 陣営と、IBM 社などを中心とする OSF 陣営が歩み寄る形で、両陣営のほとんどの有力メーカーが参加して UNIX の標準化をすすめる団体となりました。これと共に「UNIX」という商標の権利は AT&T 社から X/Open に移りました。UNIX 互換 OS が備えるべき機能を XPG として発行し、1994 年には XPG をまとめた共通 UNIX 仕様の SPEC1170 を発表しました。共通デスクトップ環境の CDE の標準化をすすめる団体でもあります。1996 年には OSF と合併して The Open Group となりました。

1.3 UNIX File System の詳細

/bin	OS 付属の最低限必要な一般的な実行ファイルを収容するディレクトリ。
/sbin	system 管理用ために最小限必要な実行ファイルを収容するディレクトリ
/etc	環境設定ファイルを収容するディレクトリ
/tmp	一時ファイルを収容するディレクトリ。
/var	各種スプール等に使用する。System log 、メイルスプールもここにある。
/usr/bin	OS 付属の一般的な実行ファイルを収容するディレクトリ。
/usr/sbin	OS 付属の system 管理、及びサーバ系の実行ファイルを収容するディレクトリ。
/usr/lib	OS 付属のライブラリを収容するディレクトリ
/usr/include	OS 付属のヘッダファイルを収容するディレクトリ
/usr/local/bin	OS に付属しない実行ファイルを収容するディレクトリ
/usr/local/sbin	OS に付属しないサーバ系実行ファイルを収容するディレクトリ
/usr/loca/lib	OS に付属しないライブラリを収容するディレクトリ
/usr/local/include	OS に付属しないヘッダファイルを収容するディレクトリ

基本的に/usr/local のディレクトリは初期の段階では存在しません。OS インストール後にアプリケーションがインストールされるときに初めて作成されます。それ以降はそのディレクトリにインストールされます。そのため/usr/local 以下のディレクトリ構成は/usr のディレクトリ構成と似たようになります。

1.4 基礎コマンド一覧

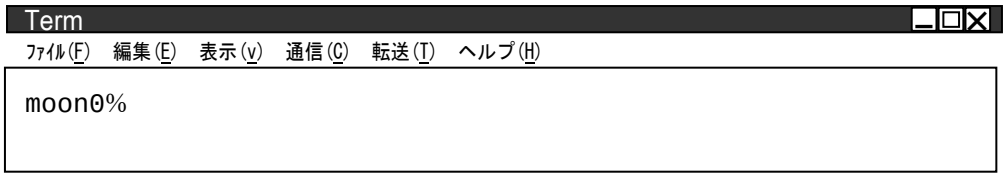
ls	filename を表示するコマンド。MS DOS の **dir** と同意
cd	Directory の移動を行なうコマンド。**Change Directory** の略語。
pwd	現在いるディレクトリ（カレントディレクトリ）の表示。知っておくと便利
cat	テキストファイルの表示(Page 制御機能無し)
more	テキストファイルの表示(Page 制御機能あり)
less	テキストファイルの表示(Page 制御機能あり)
vi	テキストエディタ
cp	ファイルをコピーするためのコマンド。
mv	ファイルを移動するためのコマンド。もしくはファイル名を変更するためのコマンド
mkdir	ディレクトリを作成するためのコマンド。
rm	ファイルの消去。もしくは、ディレクトリの消去
touch	ファイルを作成するためのコマンド。空のファイルを作ります。
exit	システムから抜けるためのコマンド。
mount	FloppyDiskDrive や CD-ROM Drive など、リムーバルメディアのマウント。
man	操作方法・マニュアルの表示

1.5 コマンド操作

コマンドはシステムプログラムの総称です。コマンドは複数のプログラムから構成されており、/usr ディレクトリのサブディレクトリに格納されています。サブディレクトリは大まかに機能別に分けられています。例えば下記に示すように分けられます。

コマンドの種類	意味
管理用のコマンド /usr/sbin	ユーザの登録など、システム管理を実施するためのコマンド群です。
一般ユーザのコマンド /usr/bin	ファイルやディレクトリを操作するためのコマンド群です。

コマンドはプロンプトに続けて入力することで実行することが可能になります。プロンプトは以下に示すような文字列で、画面上に現れます。



このプロンプトはログインの時に起動されるシェルと呼ばれるプログラムが出力します。シェルは入力されたコマンドを実行する役割を持ちます。ユーザと PC の橋渡しの役割を担い、また、カーネルを保護する役目も持っています。（シェル＝貝殻からイメージしてみてください。）

参考

各ディレクトリの役割

/etc	設定ファイル（システム管理）	/dev	デバイス
/var	システム log	/usr	Application と man（説明ファイル）
/sbin	管理用コマンド	/bin	User コマンド
/home	作業領域	/tmp	一時置き場

1.6 スーパーユーザーと一般ユーザ

■ root

システムの管理に携わる特別なユーザを、UNIX ではスーパーユーザーといいます。また、スーパーユーザーのログイン名は一般に root なので通常は root と呼ばれます。root は、どの UNIX システムにも存在する特別なユーザーアカウントです。この特別なユーザは、システムに対するフルアクセス権を持つ UNIX System での最高権限者です。基本的に何でもでき、できないことはありません。そのため、Root で操作ミスを行った場合、致命的なミスにつながることがありますので注意が必要です。基本的にアプリケーションのインストール・System の環境設定変更時以外は root で作業し内容にしましょう。

スーパーユーザーの役割には、

① システムのシャットダウン
② ユーザの登録・削除
③ ディスクの管理
④ ソフトウェアのインストール・設定
⑤ システムの監視
⑥ バックアップ

などがあります。

■ user

一般利用者権限しかもたず、その権限のおよぶ範囲内でしか作業を行なうことができません。原則として

/home/*Username*（各自の Home Directory）
/tmp
/var/tmp

以外では操作することができません。そのため、Syetem に直接影響を及ぼす作業を行なうことができないので安全です。たとえ、ミスを犯した場合も直接 System には影響しないので、いくらでもミスを犯すことができます。基本的な作業は必ず User 権限で作業を行ない、System に影響するようなミスを犯さないようにする必要があります。

1.7 実行中のユーザを変更する

たとえシステム管理者であっても、普段システムを利用するときには一般のユーザと変わりありません。普段は一般ユーザのユーザ名でシステムにログインし、システム管理者としての権限が必要になったときだけ、一時的にスーパーユーザー『root』になって作業するようにします。

一般ユーザから root になる

一般ユーザからスーパーユーザーになるには、引数なしの **su** というコマンドを使います。**su** コマンドは、あるユーザから別のユーザに切り替えるために使用します。引数としてユーザ名を指定しないと、**su** コマンドは root ユーザーアカウントに切り替えます。一般ユーザが別のユーザに切り替えようとすると、システムが切り替わるユーザのパスワードを要求してきます。

su　ユーザ名

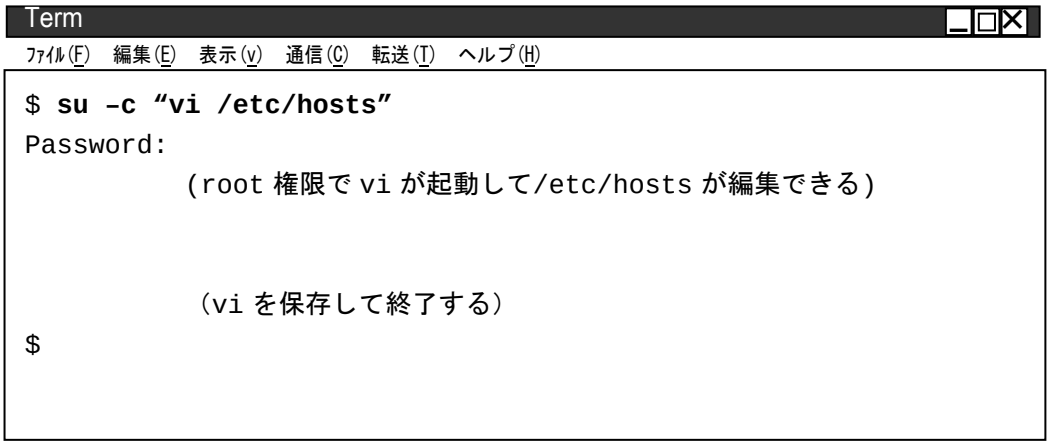
オプション	意味
-	切り替わるユーザの環境変数が読み込まれる
-c	別のユーザ権限でコマンドを実行し、終わるとすぐに元のユーザに戻る

su は Switch User の意味です。一番多い利用場面は User 権限から Root 権限への権限の移行です。移行することで User は Root 権限をもち、root と同じ作業を行なうことができるようになります。誰でも root 権限をもてるわけではなく、/etc/group を編集する必要があります。変更内容は「wheel(BSD 系の場合)」や「root(solaris の場合)」に該当 User を追加です。

```
wheel:*:0:root,hoge,ujauja          #BSD 系の場合
root::0:root,hoge,ujauja            #Solaris の場合
```

特定コマンドだけ root で実行したい

たとえば、/etc 下の設定ファイルの編集など、1 つの作業をするためだけに **su** で root になるのは面倒なとき、**su** にオプション-c をつけて指定します。そうすると root 権限でコマンドを実行し、終わるとすぐに元のユーザに戻ります。



1.8 ユーザアカウント作成とパスワード

システム管理者は、システムを利用するユーザを新しく追加したり、既存のユーザの登録を抹消することができます。

UNIX 管理上で重要なこと

1. 必ず利用者ごとにユーザアカウントを作成すること。
   システム利用者の区別がファイルやディレクトリの使用者を区別します。

2. むやみに root 権限を持つ利用者を増やさないこと。
   何でも操作可能な権限を持ったユーザが多数いるとシステム利用の方針が統一できません。

3. パスワードの管理はセキュリティ上重要
   システムに侵入され、別なユーザに成りすましてのファイルやディレクトリの操作・システムの操作が可能となってしまいます。

4. 利用者へのセキュリティに関する教育は必須
   システムにセキュリティが無ければどうなるかがわからなければ、セキュリティを守るモチベーションは得られません。セキュリティを導入した改レベルで維持していくためにユーザへのセキュリティ教育は必須です。ユーザの操作でセキュリティ上の危険が発生する行為を防ぎます。

基本的に必要なこと

1. ユーザアカウントは一意でなければならない。

2. ユーザアカウントに対応したユーザ ID（UID)を必ず設定する。これもユーザアカウント同様一意でなければならない。

3. ユーザアカウント作成時にグループ ID（GID)を設定する必要があるが、事前に GID を設定しておく必要がある。設定場所は/etc/group である。

4. コマンドライン上でユーザアカウントを作成する場合、パスワード設定はユーザアカウント時には設定しても意味はない。これは暗号化されるため、入力したものがそのままパスワードとして使用されないためである。ただ、パスワードがない状態でユーザアカウントを作成するのではなく、不明な状態で設定を行なう。例えば、パスワードを設定するフィールドに FreeBSD ならば「*（ワイルドカード）」を設定し、Solaris の場合、「**NP**」と入力する。GUI ベースの場合はその場で設定が可能である。

5. パスワードを設定する場合は、ユーザアカウント設定後、**passwd** コマンドを使用して設定する。
6. 不必要なユーザアカウントは削除すること。これによってシステムの不正使用を防止する。

新規ユーザを登録する

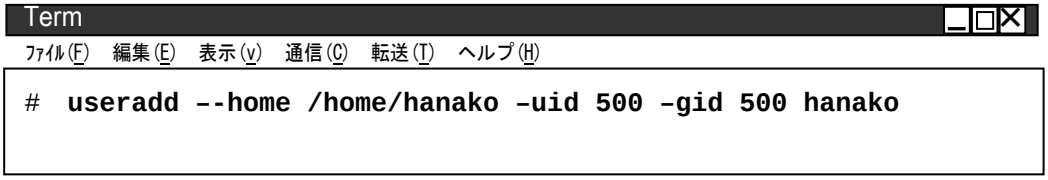
新規ユーザを登録するには、**useradd** または **adduser** コマンドを使用します。**useradd** コマンドをを利用すると、ユーザ作成と同時に/home 以下に新規ユーザ名と同じディレクトリが作成され、新規ユーザのホームディレクトリとして設定されます。

useradd　［オプション］ 新規ユーザ名

オプション	意味
--home *dir*	指定したディレクトリをホームディレクトリとする
--uid *id*	指定した UID をユーザの UID に指定する
--gid *id*	指定した GID をユーザの GID に指定する

adduser　［オプション］ 新規ユーザ名

オプション	意味
-c *comment*	パスワードファイルに入れるコメント
-d *dir*	指定したディレクトリをホームディレクトリに指定する
-g *group*	指定したグループをユーザの所属グループに指定する
-u *uid*	指定した UID をユーザの UID に指定する
-s *shell*	指定したシェルをログインシェルに指定する



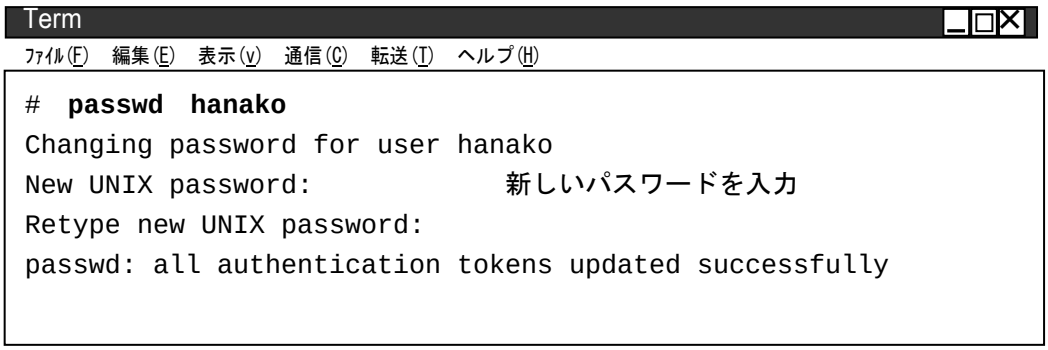
コマンドオプションの詳細は OS ごとに異なるので、マニュアルを参照してください。

新規ユーザ作成時にパスワードのロックがかかる OS があります。その場合、次節の **passwd** コマンドでパスワードを設定します。

指定したユーザのパスワードを設定する

指定したユーザのログインパスワードを変更するには、**passwd** コマンドを使用します。

passwd　　ユーザ名



前節の新規ユーザ作成時にパスワードのロックがかかる OS があります。その場合、**passwd** コマンドでパスワードを設定します。

ユーザ情報を管理する/etc/passwd ファイル

UNIX システムのユーザ情報は、一般に/etc/passwd ファイルに保存されており、一般ユーザでもファイルの内容を見ることができます。

Linux の場合、/etc/passwd ファイルの各行の書式は、コロン（：）で区切られた一連の 7 つの部分になります。

ユーザーエントリ

```
moon7：x：500：500：：/ home/moon7：/bin/bash
①　②③　④⑤⑥　　⑦
```

ユーザーエントリの各フィールドの意味

フィールド	説明
①ユーザ名	ユーザのログイン名
②パスワード	ログイン時に入力を求められるパスワードが、暗号化されている
③ユーザ ID	ユーザに割り振られる番号
④グループ ID	ユーザが所属するグループの ID 番号
⑤ユーザ情報	ユーザの氏名、オフィスの部屋番号や電話番号、自宅の電話番号など。
⑥ホームディレクトリ	ユーザのホームディレクトリのパス名
⑦ログインシェル	ユーザのログイン時に起動するシェルのパス名

システムがシャドウ・パスワードシステムを使用している場合は、/etc/shadow という別のファイルにパスワードが含まれています。（その場合、パスワードは暗号化されています。）

/etc/shadow ファイルのパーミッションは、root の読み取りアクセス権しか許可していません。

グループ情報を管理する/etc/group ファイル

UNIX システムのグループ情報は、一般に/etc/group ファイルに保存されており、一般ユーザでもファイルの内容を見ることができます。

/etc/group ファイルの各行の書式は、コロン（：）で区切られた一連の 4 つの部分になります。

```
moon7：x：500：
①        ②   ③ ④
```

ユーザーエントリの各フィールドの意味

フィールド	説明
①グループ名	グループを識別する一意な名前
②パスワード	ログイン時に入力を求められるパスワードが、暗号化されている
③グループ ID	グループ名に対応する一意な番号
④ユーザーリスト	そのグループに属するユーザ名を指定

1.9 ファイルのアクセス権限

許可の種類

ファイルの保護を行なうために、それぞれのユーザに対してパーミッション（アクセス権）を割り当てます。パーミッションには、読み取り権、書き込み権、実行権の３種類があり、それぞれ r、w、x の文字で表現します。それぞれのパーミッションの意味は以下の通りです。

記号	権利	意味
r	読み取り権	・ファイル内容の表示 ・ディレクトリの内容の表示 ・ファイル、ディレクトリのコピー
w	書き込み権	・ファイル内容の変更 ・ディレクトリへのファイルおよびディレクトリの作成 ・既存のファイルおよびディレクトリの削除
x	実行権	・ファイルの実行 ・カレントディレクトリの変更
-	許可を与えない	

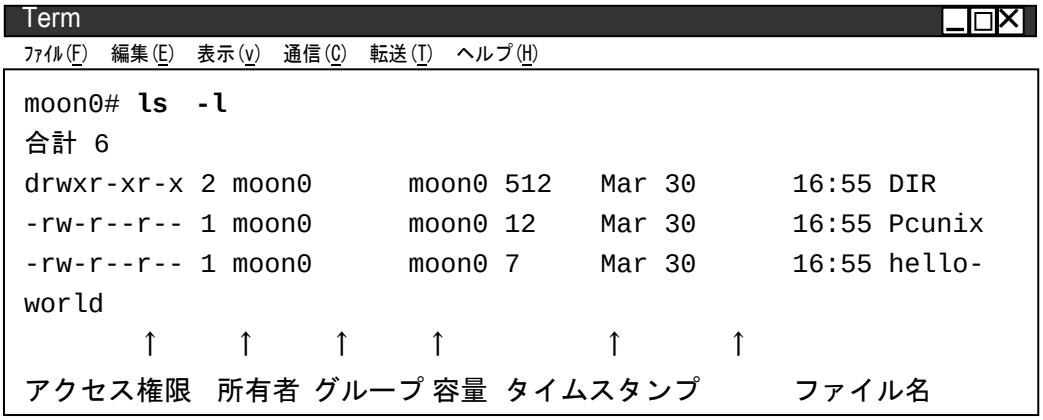
重要

ディレクトリの所有者以外に書き込み権を与えると、ディレクトリ内のファイルやディレクトリのパーミッションに関わらず、グループやその他のユーザから、それらのファイルを削除できてしまい大変危険です。これを回避するにはディレクトリにスティッキービットを設定する必要があります。

パーミッションモード

所有者、グループ、その他のユーザに割り当てられたパーミッションの組み合わせをパーミッションモードと呼びます。パーミッションモードはｒｗｘ－の組み合わせで、９つの文字で表されます。このパーミッションモードによってファイルに対するアクセス権を制御して、ファイルを保護することができます。

パーミッションモードの表示には、ls コマンドに-l のオプションをつけて使用します。



表示	意味
- または d	ファイルの種類 - はファイルを表します。 d はディレクトリを表します。
rwxr-xr-x	パーミッションモード 左から３文字ずつ、所有者、グループとその他のユーザのパーミッションを表します。 r は読み取り権を表します。 w は書き込み権を表します。 x は実行権を表します。 - は許可を与えないことを表します。
1	リンク数（ハードリンクの数）
moon0	所有者
moon0	グループ
512	サイズ
Mar 30　16:55	作成日時、最終編集日時
DIR	ファイル名またはディレクトリ名

1.10 ファイルアクセス許可の設定

ファイルに対してアクセス権を与えたり、取り消したりする時は、パーミッションモードを変更する **chmod** コマンドを使用します。通常、/bin/chmod に存在します。

chmod　パーミッションモード　　ファイル名またはディレクトリ名

パーミッションモードの指定方法にはシンボリックモードと絶対モードの２種類があります。

1.11 Network 環境設定ツール

IP Network では、IP Address の設定・経路の設定に問題がなければ、Network に接続し通信可能です。UNIX 系ではデフォルトルートを静的に設定するのが一般的です。

ifconfig

各ネットワークインターフェイスに IP Address を設定するツールです。設定内容も確認できます。

ifconfig [オプション]

オプション	意味
–a	設定内容の表示

IP Address の設定（BSD 系）

ifconfig [*network_interface*] inet [*IP_Address*] netmask [*netmask*]

コマンドオプションの詳細は OS ごとに異なるので、マニュアルを参照してください。

route

route コマンドは経路情報の設定を行なうツールです。静的経路を設定する場合に使用します。デフォルトルートさえ設定していれば通常は問題ありません。さらに先の転送先への経路設定はルータ側で行ないます。

route [オプション] [*Target_IP_Adress*] MASK [*netmask*] gw [*Router_IP_Address*]

オプション	意味
add	静的ルートの設定
delete	静的ルートの削除

コマンドオプションの詳細は OS ごとに異なるので、マニュアルを参照してください。

1.12 代表的なネットワーク調査ツール

ping

ping は IP レベル（OSI 参照モデルにおける第３層）での導通チェック用ツールで、IP パケットが相手に届いているかどうかをテストするときに利用するコマンドです。通信リンクのチェックや、特定のホストが存在し、動作しているか確認するために使用します。

ping コマンドは、相手のマシンにある種のパケット（ICMP Echo Request）を送り、それに対する返事（ICMP Echo Reply）が来るかを調べます。Unreachable ならば、ネットワークの設定や、ケーブルの配線等に問題があります。

ping［ホスト名　または　*IP Address*］

一旦実行すると定期的に **ping** を送り続けるので、Ctrl＋C キーで中断します。
実行時に-c オプションをつけて回数の指定もできます。

traceroute

traceroute コマンドは 2 台のコンピュータ間の接続を確認するために使用されます。指定したホストまで到達する途中経過を調べる役割をします。さまざまな TTL 値を使用し、接続先に ICMP エコーパケットを送信します。これにより接続先までのルートを判断し、混雑具合を調べることができます。さらに経路中のゲートウェイやルータの IP アドレスや名前の表示をします。

traceroute［ホスト名　または　*IP Address*］

指定したホストに到達できない場合、経路途中のどこまでパケットが到達しているのかがわかります。

netstat

UNIX ネットワーク環境設定の調査を行なうツールです。IP Address の設定・デフォルトルート・使用可能なサービス等々を調べることが可能です。

netstat [オプション]　[ホスト名　または　*IP Address*]

オプション	意味
–r	経路情報の表示
–in	各ネットワークインターフェイスの設定の調査
–a	使用可能なサービスの表示

コマンドオプションの詳細は OS ごとに異なるので、マニュアルを参照してください。

1.13 起動時のアプリケーション操作

起動時のアプリケーションの起動方法

FreeBSD の場合

/etc/rc.conf	デフォルトの設定の上書き
/etc/default/rc.conf	デフォルト設定用のファイル
/usr/local/etc/rc.d	Third Party 製アプリケーション起動用

初期インストール時にインストールされているアプリケーションはここで UNIX 起動時の設定を行ないます。デフォルトの設定の/etc/default/rc.conf を編集すれば起動時の設定変更は可能ですが、このファイルは変更せずに変更する内容を行ごと/etc/rc.conf に書き込めば、起動時に上書きされます。

1.14 Inetd 経由のアプリケーションの操作

inetd とは各種サーバアプリケーションの接続管理ツールのことです。必要なサーバアプリケーションを常時起動しておくとメモリ等のサーバ資源を常時消費してしまいます。普段使ってもいないアプリケーションを立ち上げておくのは、資源の浪費になるので、inetd が一括して管理を行ないます。基本的動作は、必要に応じて接続要求に対応したサーバアプリケーションを起動します。利点としてはメモリ資源を消費しないということですが、欠点は inetd が止まればそのコントロール下にあるサーバアプリケーションはすべて使用できなくなります。

inetd のコントロール下に入るサーバーアプリケーションは/etc/inetd.conf で制御します。設定されたサーバアプリケーションの PID は接続されない限り存在しません。しかし、ポートは開いている状態となります。（**netstat** –a で確認できます。）

基本的な inetd 対応サーバアプリケーション

- telnet
- ftp
- tftp
- pop3
- imap4
- netbios-ssn

2章　初期設定

2 初期設定

2.1 利用計画

サーバを構築する前に考えなければいけないこと

- どんなサービスを行なうのか？
- HTTP、SMTP、POP、DNS、RADIUS 等々
- サービスの種類によって、機器の構成と設定内容が異なる。
- Mai l Server を構築する場合、Mail Spool を大きめに用意する必要がある。
- DNS Server を構築する場合、特に専用に用意するものはない
- Web Server を構築する場合、利用者の Home Directory を多めに用意必要がある。
- バックアップする方法が異なってくる
- サービスの規模は？
- 機器構成が異なる。
- 機器のスペックはが異なってくる。
- 機器の必要台数が異なってくる。
- 運用体制が異なってくる。
- ネットワーク構成は？
- トラフィック量によりネットワーク構成を考えなければいけない
- 機器構成は？
- 冗長構成にするのか？
- 機器構成に問題はないのか？
- RAID 構成するのか
- CPU スピードは？
- Memory の容量は？
- HDD の容量は？
- NIC は？
- System 構成は？
- OS はどれにする？
- パーティションの切り方は？

2.2 運用方法

機器構成によって運用方法はことなります。

機器数は少数であるが、多機能のサーバを用意した場合

利点

① 管理対象サーバ・機器数が減るので、管理は楽。

欠点

① １つのサーバで多種多様な障害が発生した場合に、障害の切り分けがしにくい。
② ひとつのサービスが障害を起こして停止すると他のサービスに影響が出る場合があるので、障害を起こさせないような構成にする必要がある。
③ １つのサーバにかかる負荷が高くなるので、障害が発生しやすくなる。

備考

機器数が少ないからといって、必ずしも機器コストが安いということはありません。機器数が少なくなればそれだけその機器１台が重要になり、停止しないシステム構成にする必要がでてきます。また、機器数が少なくなれば、それだけの負荷に耐えられるだけの機器構成が必要になり、結果的に機器のコストがかかる場合があります。

単機能でサーバを構築し、多数機器を用意した場合

利点

① １台のサーバで障害が発生しても、影響が少なくてすむ

② 障害が発生しても、多機能サーバと比較して原因を追求しやすい。

③ 障害が発生しても復旧が容易。

欠点

① 管理対象サーバ数が多くなると、管理負担も増加する。

2.3 セキュリティ

サーバは動けばいいというものではない。

サーバは安定し、安全に運用されなければいけません。

障害のもとになる事象

·機器障害（ネットワーク障害も機器障害）には、

→ 事前検知で対応します。

→ 壊れない機械はありませんから、避けられない障害です。

→ 機器構成内容で対応します。

·外部からのクラック・Dos 攻撃は、

→ 「いたちごっこ」です。

→ 出来得る限りセキュリティの高いシステム構築を行い対応します。

·サーバがクラックされた場合には、

→ サービスが停止します。
理由は侵入者によるシステムダウンです。

→ 侵入されたシステムは OS からインストールのしなおしをする必要があります。
どこにわなを仕掛けられているのかを調査することが困難です。

→ 踏み台にされる可能性があります。
他のサーバに影響を及ぼす可能性があります。
他人のサーバに影響を及ぼす可能性があります。
あなたの信用がなくなります。

2.4 セキュリティ対策

クラックされないためには

- セキュリティ情報には気を配りましょう。
  → 各 OS・アプリケーションにセキュリティホールがあった場合には、ホームページ・メイル等でアナウンスがありますから、それに注意しましょう。

  → もしセキュリティホールがありその機能を使っているのならば、使用を停止するか、Patch をあてるなどをして対策をとる必要があります。

- 使用しないサーバアプリケーションは停止しましょう。
  → 普段使用していないサーバアプリケーションを起動しておくことは、立ち上がっていることを忘れてしまうことになり、セキュリティホールが発見されていたとしても注意をはらわない可能性があります。またのため必要のないサーバアプリケーションは停止すべきです。

- アクセスリスト等でサーバにアクセスできる IP Address・利用者を制限しましょう。
  → すべてに対処が出来るわけではないので注意が必要です。

  → 無秩序に使えるようにするのはきわめて危険です。必要に応じて制御すべきです。

  → 接続できなければ侵入されることもありません。

- サーバアプリケーションの特質に応じた対応が必要です。
  → サーバアプリケーションによっては無手順（認証無し）で利用できるものもあり、そのようなサーバアプリケーションには十分に注意しましょう（tftp 等）。

- 通信を暗号化します。
  → インターネット越しに通信を行なう場合、できる限りパケットは暗号化すべきです。

  → 暗号化されていない状態でパスワードを入力するのはできる限り避けるべきです。

2.5 セキュリティ対策の問題点

セキュリティ対策は重要ですが、セキュリティ対策を施せば施すほど利用しずらくなります。セキュリティと利用のしやすさを天秤にかける必要があります。もっともよいセキュリティ対策は利用者にさとられないようにかけるべきものですが、実際には不可能です。

構築方法の基本

インストールしたてのシステムの状態

■ セキュリティレベル的にいえば低い状態です。

→ 不必要なサーバアプリケーションが起動しています。

→ アクセスリスト等で制限が加えられていはいません。

■ サーバアプリケーションにセキュリティホールがある可能性があります。

→ インストールディスクが作成されてから、実際にインストールされるまでにセキュリティホールが発見されていることもあります。

構築方法の基本手順

① ネットワーク的に安全なところで OS をインストールします。この「ネットワーク的に安全なところ」とはプライベートセグメントなど、外部から侵入されないと思われる場所ということです。例えば、ローカルセグメントです。最も安全なのはネットワークから切り離すことです。

② 必要なサーバアプリケーション以外のすべてのサーバアプリケーションを停止します。何を停止するべきかは OS によってことなるものの、実際的に「syslogd」以外ほとんど必要ありません。

③ OS のシステムに対するセキュリティホールの修正を行ないます。この状態でもっとも安全な状態となります。

④ セキュリティ対策を行ってから、１つづつ必要なサーバアプリケーションを起動します。例えば、tcp_wrapper と openssl、openssh 程度を使えるようにします。

⑤ サードパーティ製のサーバアプリケーションをインストールし環境設定を行ないます。

⑥ 動作確認後、インターネットからアクセスが可能なセグメントに設置します。

⑦ 最終設定を行い、試験運用を行ないます。

⑧ 問題がなければ運用を開始します。

構築する前に

実際に OS のインストール・アプリケーションの構築を行なう前に付属のドキュメントには全て目を通しましょう。インストール方法等で重要事項の記載があり、必ず目を通します。重要なのは、

① README
② INSTALL

といった感じのファイルです。

ここ最近の悪い風潮として、①Web で検索し、②そこに書かれている方法を使用して構築する人が多いことです。そこにかかれている方法が正しい方法であるという保証はどこにもなく、基本的にその方法は良い方法ではありません。あくまでも参考程度におさえておきましょう。信じていいホームページはそのアプリケーションを開発しているところのホームページのみです。

余談

基本的にこれらのドキュメントは英語でかかれています。ほとんど日本語でかかれているものはありません（日本人が作成したのでさえ、日本語ドキュメントがないものすらあります）。しかし、きわめて簡単な英語ですから（文法的には中学レベル）、読めるようにはなってください。

構築後は

サーバの運用を開始できたからといって、現在の設定を今後もずっと続けられることはありません。管理するユーザー数・ホスト数の増加に伴い、サーバのスペックをアップグレードするだけでなく、新たなセキュリティホールへの対処など、作業が発生します。

2.6 余談（環境構築の際のトラブルを避けるための知恵）

システムの時刻あわせがなぜ重要なのか

システムの時刻あわせがなぜ重要なのでしょうか

① 障害発生時刻を特定します。

障害情報等は基本的に log に書き出されますが、そこに記載されているタイムスタンプはシステムに設定されている時刻です。もし実際の時刻と異なる場合、書かれた障害情報が特定できない可能性があります。

② タイムスタンプを見ているアプリケーションがあります。

例えば、かつて作成されたファイルは時計が正しければ古いタイムスタンプが押されています。しかし時間がズレている場合、現時刻より未来のタイムスタンプが押されていることもあり得ます。この時にアプリケーションでタイムスタンプを見ているとエラーを出し、最悪の場合、機能が停止します。

時刻合わせ方法

初期状態のシステムの時刻合わせ方法	→	BIOS に設定されている時刻
ネットワーク上での時刻合わせ方法	→	NTP を使用

基本的な時間の合わせ方

NTP を利用して、時刻を合わせるべきです。

BIOS によるシステムの時刻の設定は起動時のみですから、それ以降は OS 側でコントロールされます。たとえ最初に BIOS の時刻があっていたとしてもじにズレが生じます。NTP を利用すると自動的に NTP Server と同期をとるため、正確な時間をいつで刻むことが出来ます。最近では GPS を用いた NTP Server も存在しているのできわめて正確です。

3章　DNS 構築

3 DNS 構築

3.1 DNS とは？

DNS

TCP/IP の通信では、ネットワークに接続されているコンピュータを識別するために、必ず各マシンに IP アドレスを設定し、その IP アドレスを基に通信を行ないます。しかし、実際に任意の相手先と通信を行なう際に IP アドレスを使用するのはとても不便です。そこで、インターネットに接続しているネットワークそれぞれに組織名（ドメイン名）をあたえ、ユーザーにとって判りやすい系統的な管理の方法が考えられました。「ホスト名＋ドメイン名」といった場所指定を全部表記する方法を FQDN（Fully Qualified Domain Name）といいます。

UNIX システムでは、ホスト名と IP アドレスの対応をつけるために、/etc/hosts というファイルにホスト名と IP アドレスの対応表を書いておいて、この表を見ながらホスト名を IP アドレスに変換するということが行われていました。この方法には、

① 管理対象のホスト数が増えるにしたがって HOSTS ファイルのメンテナンスと配布に負担がかかる、

② インターネット上の全ホストのエントリを HOSTS ファイルに反映することは不可能、

という弱点があります。そこで HOSTS ファイルのエントリをデータベースとして集中管理する方法が考えられました。そこで、使用されているのが Domain Name System（DNS）です。DNS とは、ホスト名から IP アドレスを、またはその逆に IP アドレスからホスト名を探すためのシステムです。たとえば、Web ブラウザで入力する URL や電子メイルアドレスは、DNS によって IP アドレスに変換されて目的のホストマシンが特定されます。このホスト名と IP アドレスの関連付けサービスを提供しているサーバをネームサーバと呼びます。

DNS は DNS サーバ自身がインターネット上の全ホストのエントリを管理する必要はなく、ドメインのツリー構造を利用し、名前解決の問い合わせはターゲットのホストを管理している DNS サーバに問い合わせるシステムです。

アプリケーション的に有名なものとしては、BIND、NIS があります。

BIND

BIND とは、ネームサーバで使用されるプログラムです。ネームサーバで使用されるプログラムには UNIX 系の BIND、マイクロソフトの DNS マネージャ、アップルの MACDNS などがありますが、一般的には、BIND が主流になっています。

BIND は、Ver4.x、Ver8.x、Ver9.x　と大きく３つに分かれています。現在では、Ver8.x が一番使われています。Ver9.は、IPv6 に対応しています。

NIS

Sun Microsystems 社が開発した、ネットワーク上の複数のコンピュータ間でユーザ情報を共有するシステムです。ホスト名解決機能を持っています。

3.2 DNS の仕組み

DNS は DNS サーバ自身がインターネット上の全ホストのエントリを管理する必要はなく、ドメインの階層構造を利用し、名前解決の問い合わせは問い合わせをしたいホストを管理している DNS サーバに問い合わせるシステムです。

DNS はその階層構造でデータベースを分散管理しています。それぞれの階層は、ひとつ下の階層のデータを持っているので、上の階層から順番に下の階層へと調べていけばよいということになります。分散して管理する範囲をゾーンと呼び、階層構造になっているドメイン名を管理しやすい単位（ゾーン）に分割しています。

DNS では、DNS サーバであるネームサーバと DNS クライアントであるリゾルバがあります。

ネームサーバ

リゾルバからの要求によりドメイン名から IP アドレスへの変換を行ない、結果をリゾルバに返すサーバ。ある階層のドメイン名と IP アドレスが対応付けられたデータベースを持ち、これを参照して変換を行なう。

ルートネームサーバ

TLD を管理するネームサーバで、階層構造の一番上にある。ルートサーバは 13 台（a.root-servers.net～m.root-servers.net）用意されていて、世界中に分散配置されている。日本にも 1 台（m.root-servers.net）ある。

リゾルバ

WWW ブラウザなどのアプリケーションに代わって、ネームサーバに問い合わせるクライアント型プログラム。

名前解決

IP アドレスとドメイン名の対応付けのことを、名前解決といいます。ドメイン名から IP アドレスの名前解決を「正引き」といい、IP アドレスからドメイン名の名前解決を「逆引き」といいます。

キャッシュ

DNS は、一度問い合わせた情報をキャッシュに保存し、ルートサーバへの問い合わせを減らし、名前解決を高速化します。

3.3 BIND8 の構成

BIND は次のファイルから構成されます。

named

BIND の本体。

named.conf

BIND のコンフィギュレーションファイル。BIND4 の場合は、named.root です。

root.cache

インターネットのルートとなる DNS を記述したファイルです。現在、A から M の 13 個のサーバがあります。ファイル名は任意に与えられます。named.conf 内の記述と合わせます。

正引き用 zone ファイル

ホスト名から IP Address を参照する際に使用されるデータファイルです。named.conf 内の記述と合わせます。

逆引き用 zone ファイル

IP Address からホスト名を参照する際に使用されるファイルです。named.conf 内の記述と合わせます。

ローカルホスト情報ファイル

localhost の IPAddress(127.0.0.1)からホスト名（localhost）を参照する際に使用するファイルです。

named.conf

Named.conf の設定

```
options {                          ←BIND のオプション設定をする場合に使用する。
directory   “/etc/namedb”;         ←各 ZONE(ドメイン情報)を収容するデイレクトリを指定する。
      };                           ←option の行の終わり

zone “.” {                         ←キャッシュゾーンの設定
      type hint;                   ←この ZONE の種類
file “root.cache”;                 ←キャッシュファイルのファイル名
      };                           ←キャッシュゾーンの設定の終わり
zone “0.0.127.in-addr.arpa” {      ←localhost 用の逆引き用の設定
type master;                       ←ZONE が Primary か Secondary かを設定。master が Primary で、slave が Secondary。
file “primary/localhost.rev”;      ← localhost 用の逆引き用のファイル名。
      }                            ← localhost 用の逆引き用の設定の終わり;

zone “ico-g.net” {                 ←正引きゾーンの設定（例  ico-g.net)
type master;
file “primary/db.ico-g.net”;       ←正引きゾーン用のファイル名
      };                           ←正引きゾーン用の設定終わり

zone “152h.22.204.61.in-addr.arpa” {←正引きに対する逆引きゾーンの設定
type master;
file "primary/db.152.22.204.61";
      };                           ←逆引きゾーンの設定
```

注意点

各行の最後に「;」が必要です。

Secondary Name Server の運用

Secondary Name Server とは？

DNS の冗長構成を行なうための仕組みです。Primary にアクセスし zone 情報を取得し同期をとります。システム的には Primary server と Secondary Server が存在していますが、インターネット的には区別ありません。インターネットから見たときにはどちらも同様に見えます。

設定方法

同期を取る必要のある ZONE に対して設定を行ないます。

```
zone “ico-g.net” {                          ←正引き用の ZONE の設定
      type slave;                           ←Secondary の設定
      file “ico-g/db.ico-g.com”;            ←取得した ZONE 情報を収容するファイ
                                            ル。ファイルは自動的に作成される。
      masters {                             ←Primary  DNS の場所の設定
192.168.2.103;
            };                              ←そのおわり
            };

zone “64/27.161.134.210.in-addr.arpa” {     ←逆引き用の ZONE の設定
      type slave;                           ←Secondary の設定
      file “ico-g/db.64.161.134.210”;       ←取得した ZONE 情報を収容するファイ
                                            ル。ファイルは自動的に作成される。
      masters {
      192.168.2.103;
      };
      };
```

Localhost 用の逆引きゾーン

このファイルは 127.0.0.1 というアドレスを localhost の名前に変換する際に使用されます。

ループバックファイル /var/named/named.local

```
$TTL  86400
@     IN    SOA         localhost   .     root.localhost.   (
                              1999022703  ;Serial
                        28800       ;Refresh
                        14400       ;Retry
                        3600000     ;Expire
                        86400 )     ;Minimun
      IN    NS          localhost.
1     IN    PTR         localhost.
```

正引きゾーン

ゾーン正引きファイルは、担当ゾーンにおいてホスト名やドメイン名から IP アドレスを検索するためのデータベースファイルです。

正引きゾーンファイル /var/named/named.classroom.com

```
$TTL  86400
@     IN    SOA   moon1.classroom.com.    hostmasster.classroom.com.
      (
                        1999022703  ;Serial
                        28800       ;Refresh
                        14400       ;Retry
                        3600000           ;Expire
                        86400 )           ;Minimun
IN    NS                moon1
IN    MX    10          moon1

moon1 IN    A           192.168.0.1
moon2 IN    A           192.168.0.2
www   IN    CNAME moon1
```

■ **オリジン**

先頭の@は設定ファイルで定義したドメイン名を表し、デフォルトのカレントオリジンを表します。

■ **シリアル番号**

ゾーン情報が更新されたかどうかはこの番号で判断されます。

■ **リフレッシュ間隔**

セカンダリ・ネームサーバーが、このデータベースファイルが変更されているかどうかをチェックする間隔です。秒単位で指定します。

■ **再試行間隔**

セカンダリ・ネームサーバーが、このデータベースファイルのコピー（ゾーン転送）に失敗したときに、再試行を行うまでの間隔です。秒単位で指定します。

■ **情報破棄時間**

セカンダリ・ネームサーバーが、ゾーン情報のダウンロードを試み続ける時間です。この時間が過ぎると、古いゾーン情報が破棄されます。秒単位で指定します。

■ **情報有効期間**

ネームサーバが、このデータベースファイルのリソースレコードのキャッシングを行うことが許される時間です。秒単位で指定します。

Bind 9 から w(week)・d(day)・h(hour)・m(minuets)・s(second)いづれでも指定出来るようになりました。

逆引きレコード定義ファイル

このファイルは、IP アドレスをホスト名へ変換するための情報を定義します。

逆引きゾーンファイル /var/named/named.192.168.0

```
$TTL  86400
@     IN    SOA   moon1.classroom.com.
      hostmaster.classroom.com.     (
                        1999022703  ;Serial
                        28800       ;Refresh
                        14400       ;Retry
                        3600000     ;Expire
                        86400 )     ;Minimun
      IN    NS          moon1.classroom.com.
1     IN    PTR         moon1.classroom.com.
2     IN    PTR         moon2.classroom.com.
```

3.4 DNS と Mail

MX とは？

Mail Exchange の略で、DNS のソーンファイル内のメイルサーバエントリを指定します。メイルアドレスから送るべきホストを決定するために使用されます。実際に、メイルアドレスはホスト名ではなくドメイン名が使用されています。しかし、ドメイン名のみでは通信はできず、必ずホスト名もしくは、IP Address を必要とします。MX レコードはドメイン名に対するメイルの送り先ホストを指定します。

```
ico-g.net         IN    MX    0     mail1.ico-g.net.
```

メイルはインターネットの中で主要な機能です。可能な限り停止する時間を少なくするべきです。しかし、障害等で使用できない場合も少なくはありません。その対策として冗長構成をとることが可能です。これは MailServer 側で行なうのではなく DNS 側で設定します。一つのドメインに対して複数の Mail Server を用意ができます。Primary Mail Server・Secondary Mail Server の区別は Preference 値（重み）を設定で行ないます。数が小さいものを Primary Mail Server として優先して接続を試みます。接続できない場合次に小さな値の Mail Server にメイルを届けます。数字そのものに厳密的な意味はありません。

```
ico-g.net         IN    MX    0     mail1.ico-g.net.
                  IN    MX    100   mail2.ico-g.net.
```

ドメインでなくホストに直接メイルを送ることが多々あります。その対策として、ホスト名に対しても MX を設定すべきです。

```
ujauja.ico-g.net.       IN    A     192.168.3.10
                        IN    MX    0     ujauja.ico-g.net.
hoge.ico-g.net          IN    A     192.168.3.11
                        IN    MX    0     hoge.ico-g.net.
```

3.5 構築

BIND8 の構築

```
> make depend
> make
> su
# make install
```

注意

FreeBSD 等の Free UNIX 系の OS の場合には、BIND は標準でインストールされています。セキュリティホール等が存在するという場合でない限り、それを使用しても問題はないでしょう。

Free UNIX で標準インストールされている BIND は、多くが/etc/namedb のディレクトリに named.conf が収容されていますが、それ以外の場合（自分でコンパイルする等）には基本的に named.conf は/etc に置くようにします。/etc/namedb に置くように設定する場合は、

① named の起動時にオプションを付ける

```
named -b /etc/namedb/named.conf
```

② コンパイル時に設定を変更する

などの方法があります。①がお薦め。②は豊富な経験が必要です。

UNIX System では DNS を早めに立ち上げる必要があります。例えば、Qmail 等では起動時に MX の参照を行ないます。DNS が起動していないと MX 参照ができないので Qmail が正常に起動しません。

Sndmail の場合では FQDN を必要とします。これは/etc/hosts にエントリを記載することで代用可能です。

3.6 BIND の起動

起動

```
# /usr/local/sbin/named -b /etc/namedb/named.conf
```

基本的にオプションは必要ありませんが、どのコンフィギュレーションファイルを使用しているのか明示的に指定するのでわかりやすくなります。デフォルトでは「/etc/named.conf」、もしくは「/etc/namedb/named.conf」ですので、可能な限り指定するとトラブルが少なくなります。オプションは「-c」でも同様です。

立ち上げ時の注意

Syslog は必ずチェックしましょう。各設定ファイルで問題があると正常に動作しません。Syslog にはその情報が書き込まれるので必ずチェックし、間違いを修正します。問題がなければ、

```
Nov 2 10:03:52 ujauja named[162]:[ID 295310 daemon.notice] Ready to answer queries.
```

だけが書き込まれているはずです。問題がある場合には「どこが間違っているのか」まで記載されています。

Zone 情報変更時の注意

各ゾーンの情報を変更した時には、必ず ZONE の Serial の番号を大きくしましょう。大きくしなければリロードされません。

3.7 BIND の自動起動

FreeBSD の場合

/etc/rc.conf への追加

```
named_enable="YES"                          ←named を使用する
named_program="/usr/sbin/named"             ←named の場所（Full Path）
named_flags="-b /etc/namedb/named.conf"     ←オプションの設定（基本的に必要ないが書いておいたほうが無難）
```

Solaris8 の場合

/etc/rc2.d/S72inetsvc

変更前

```
#if [ -f /usr/sbin/in.named -a -f /etc/named.conf ]; then
#       echo 'starting internet domain name server.'
#       /usr/sbin/in.named &
#fi
```

変更後(各自の好みにより設定)

```
if [ -f /usr/local/sbin/named -a -f /etc/namedb/named.conf ]; then
      echo 'starting internet domain name server.'
      /usr/local/sbin/named –b /etc/namedb/named.conf&
fi
```

3.8 Nslookup

DNS の動作を確認するには、nslookup コマンドを使います。nslookup コマンドは、UNIX にも WindowsNT にも標準で用意されています。nslookup を起動すると、参照するネームサーバを表示して入力待ちとなります。そこでは、次のようなコマンドが使用できます。

実行例

```
# nslookup
Default Server:   moon1.classroom.com
Address:    192.168.0.200
#
```

実行例 自分のホスト名を入れてみます

```
> moon1.classroom.com
Server:     moon1.classroom.com
Address:    192.168.0.200

Name:       moon1.classroom.com
Address:    192.168.0.200
>
```

プロンプトに対して名前（ドメイン名）を入力すると、それに対応するリソースを表示します。

実行例 逆に IP アドレスを入力してみます

```
> 192.168.0.1
Server:     moon1.classroom.com
Address:    192.168.0.200

Name:       moon1.classroom.com
Address:    192.168.0.200
```

オプションコマンドの実行例

```
#nslookup
Default Server:  ns1.ico-g.co.jp
Address:  192.168.0.2

> server 192.168.0.3
Default Server:  ns2.ico-g.co.jp
Address:  192.168.0.3

> set type=MX
> ico-g.co.jp
Server:  ns2.ico-g.co.jp
Address:  192.168.0.3

ico-g.co.jp MX preference = 10, mail exchanger = smetana.ico-g.co.jp
smetana.ico-g.co.jp     internet address = 192.168.0.7
> set type=NS
> ico-g.co.jp
Server:  ns2.ico-g.co.jp
Address:  192.168.0.3

ico-g.co.jp       nameserver = falla.ico-g.co.jp
ico-g.co.jp       nameserver = bartok.ico-g.co.jp
ico-g.co.jp       nameserver = weber.ico-g.co.jp
verdi.ico-g.co.jp       internet address = 192.168.1.2
falla.ico-g.co.jp       internet address = 192.168.0.3
bartok.ico-g.co.jp      internet address = 192.168.0.2
weber.ico-g.co.jp       internet address = 192.168.2.2
> exit
#
```

server コマンドでクエリする任意のネームサーバを選択し、**set** コマンドでクエリするレコードの種類を設定すると、ターゲットのドメイン名を入力に対し一致したレコードを表示できます。

3.9 DNS 管理の委任（親子）

DNS はドメインネーム空間の階層構造を分散管理するシステムですから、任意のネームサーバは任意のレイヤーの任意の空間のみを管理します。上位の空間は親のネームサーバにクエリして解決します。下位のネーム空間についても、全てを管理することが現実的でない大きさの場合や、管理負担を分散するために、子のネームサーバ管理権限を委任します。

典型的な管理委任モデルは、ルートサーバーが第２レベルのネームサーバを紹介するにとどめ、全世界のホストを管理していない構造です。

```
.                       3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
（以下略）
```

委任の方法（正引きの場合）

設定例

```
;example.co.jp.のゾーンデータ"example.db"
example.co.jp.    IN    SOA   ns.example.co.jp. Admin.example.co.jp.(
2001103001   ;
10800        ;
3600         ;
604800)      ;

;ネームサーバ(NS)
example.co.jp.    IN    NS    ns.example.co.jp.

;ドメイン名に対応する IP アドレス（A）
ns    IN    A     192.168.1.2
www   IN    A     192.168.1.3
mail  IN    A     192.168.1.4

;メールに使用する MX レコード（MX）
example.co.jp.    IN    MX    10    mail.example.co.jp.

;tech ゾーンの委任先
tech        86400 IN    NS    ns.tech.example.co.jp.
ns.tech     86400 IN    A     192.168.2.2
```

/24 より小さいサブネットの DNS 管理

問題点

固定 IP Address を割り振る専用線サービスの場合（OCN 等）では、利用者に/29（29bit マスク）を割り振ります。現在では IP Address の枯渇が叫ばれているため、少数のネットワーク機器しか接続しない場合には、/24 の IP Address 群は配布されることはありません。このときに、正引きゾーンの解決はサブネットに依存しないので管理上問題はありませんが、逆引きゾーン解決の場合は基本的に/24 単位で管理される必要があるため問題が生じます。可能な限り正引きと逆引きの結果を合わせる必要があるため（ISP 側で管理を行なえば問題はないが）、設定の変更等が自由にできない可能性があります。

解決方法

例　　92.168.2.0/24 のネットワークを 192.168.2.0/29、192.168.2.8/29…に分割して、それぞれのサブネットを各サブネット管理者で管理する場合

親 DNS

```
1.2.168.192.in-addr.arpa.      IN    CNAME 1.0-7.2.168.192.in-addr.arpa.
2.2.168.192.in-addr.aepa.      IN    CNAME 2.0-7.2.168.192.in-addr.arpa.
                               ：（略）
0-7.2.168.192.in-addr.arpa.    86400 IN NS ns1.ujauja.com
0-7.2.168.192.in-addr.arpa.    86400 IN NS ns2.ujauja.com

9.2.168.192.in-addr.arpa.      IN    CNAME 9.8-15.2.168.192.in-addr.arpa.
10.2.168.192.in-addr.aepa      IN    CNAME 0.8-15.2.168.192.in-addr.arpa.
                               ：（略）
9-15.2.168.192.in-addr.arpa.   86400 IN    NS    ns1.hogehoge.com.
0-7.2.168.192.in-addr.arpa.    86400 IN    NS    ns2.hogehoge.com.
```

ujauja.com の named.conf の逆引きゾーン

```
zone “0-7.2.168.192.in-addr.arpa" {
type master;
file "primary/db.0-7.2.168.192";
            };
```

db.0-7.2.168.192 の中身

```
$TTL 86400
@ IN SOA ns1.ujauja.com. root.ns1.ujauja.com. (

2001091901        ; Serial
10800             ; Refresh
3600              ; Retry
604800            ; Expire
86400       )     ; Minimum
IN    NS    ns1.ujauja.com.
IN    NS    ns2.ujauja.com.
1     IN    PTR   router.ujauja.com.
2     IN    PTR   host1.ujauja.com.
3     IN    PTR   ns1.ujauja.com.
4     IN    PTR   ns2.ujauja.com.
```

3.10 アプリケーションと DNS

逆引きできないネットワーク機器のアクセス制限

問題点

最近のネットワークコマンドやアプリケーションでは逆引きを行なう場合が多くなっています。逆引きが出来ないからといってそのコマンドが使えないというわけではありませんが、逆引きが正常に行われない場合、DNS 参照のタイムアウトまで待機しなければなりません。これはセキュリティの観点から逆引きできないネットワーク機器からのアクセスを制限するための機能です。

回避方法

この現象は逆引きを設定すれば解決できます。このケースが問題となるのは、Private IP Address を使用した場合です。Global IP Address の場合は逆引きを設定するので問題はありません。

オフィス LAN 等では、ISP から割り当てられている IP Address 数が実際に使用するネットワーク機器よりも少ないため、NAT 等を使用しています。Private 側のセグメントは当然 Private IP Address を使用することになりますが、その Private IP Address もきちんと DNS に登録すべきです。

Private Segment に対する hostname 照会のアクセス制御

問題点

Private Segment はインターネット側からアクセスする必要はない（Private Segment はインターンネットからアクセスできないという理由もあります）にもかかわらず、DNS はたとえ Private Segment に対する hostname の照会でもアクセスできる限り返事を返してしまいます。これはあまりいいことではありません。

回避方法

これはアクセスできるセグメントを制御すべきである。

```
zone "peace.ico-g.com" {
      type master;
      file "db.peace.ico-g";
      allow-query {     10.135.167.0/24;
                        210.134.161.64/27;
                  };
      allow-transfer {  10.135.167.0/24;
                        210.134.161.64/27;
                        };
};
```

```
zone "167.135.10.IN-ADDR.ARPA" {
      type master;
      file "db.167.135.10";
      allow-query {     10.135.167.0/24;
                        210.134.161.64/27;
                  };
      allow-transfer {  10.135.167.0/24;
                        210.134.161.64/27;
                        };
};
```

3.11 Version UP

Version UP のタイミング

① セキュリティホールが見つかった時
② パフォーマンスを上げたい時（必ずしも意味があるわけではありません）

Version UP の方法

① BIND は再起動しない限り、named を書き換えても問題は出ません。
② 再起動している瞬間は使用できませんが、瞬断です。
③ 再起動の場合は、HUNGUP させればいい。

4章　mail 環境構築（sendmail 編）

4 mail 環境構築（sendmail 編）

4.1 Sendmail とは？

sendmail は 1982 年にアメリカの Eric Allman 氏によって開発された電子メールサーバソフトウェア(MTA：Message Transfer Agent)です。ユーザが送信したメイルを受け取って、他のサーバと連携してバケツリレー式に目的地まで配送するメイル配送ソフト（MTA）の代表的なものです。 ほとんどの UNIX には、標準添付されています。

最新バージョン

バージョン 8 の sendmail の場合、4 つのリリースツリーがあります。

何らかの理由によって、バージョン 8.12.0 にアップグレードできない、またはアップグレードしたくない場合は、バージョン 8.11.6 や 8.10 と 8.9 の sendmail を利用できます。バージョン 8.9 の最後のバージョンは 8.9.3 で、8.10 の最後のバージョンは 8.10.2 です。

機能

SMTP はインターネットにおける代表的なメイル転送プロトコルです。しかし SMTP にはクライアントを認証する手段が用意されていなかったため，適切な設定を行っていないと，いわゆる SPAM メイルの中継や「なりすまし」メイルの原因となります。

これに対抗するため、「POP before SMTP」と呼ばれる認証方式が確立されました。POP before SMTP では、メイルを受信する際に使われる POP3 の認証を利用し、認証が行われた IP アドレスから時間を限定して SMTP によるメイルの送信を許可するという仕組みです。一定時間の経過後は認証が消失してしまうため、再び POP3 でアクセスして認証作業を行わなければなりません。しかし、POP before SMTP はすでに普及しているサーバやクライアントで認証を実現できるというメリットがありました。SMTP Authentication までのつなぎとして利用できます。

SMTP Authentication ではサーバとクライアントの双方が対応していなければならないものの、メイル受信の際に SMTP サーバとユーザとの間でユーザアカウントとパスワードの認証を行い、認証された場合のみメイルの送信を許可します。「AUTH-LOGIN」「AUTH PLAIN」「AUTH CRAM-MD5」などいくつかのユーザ認証方式があります。このうち「AUTH CRAM-MD5」は、パスワード文字列がそのままネットワークを流れることがないように、暗号化が施されます。

4.2 Sendmail の構築

Sendmail の構築方法

```
>cd  $WORK_DIRECTORY
>/bin/sh Build
```

特に注意するところはありません。

Sendmail は基本的に標準で OS に付属しています。

→ 新規インストールの必要はないように見えます。

→ 標準添付の Sendmail を使用しても問題はありません。

→ 使用する環境設定ファイル（sendmail.cf）がオリジナルと異なる時に問題となる場合があります。

→ 基本的に WIDE CF は使用できません。

→ 設定変更をしないのならばそのまま使用して問題はありませんが、変更する場合に負担が生じます。

→ ほとんど設定変更するでしょうから、インストールしなおしたほうが無難です。

4.3 バージョン問題

では、どのバージョンのをインストールするのか？について考えましょう。

FTP Server にあるバージョン

sendmail-8.9.3
sendmail-8.10.2
sendmail-8.11.6
sendmail-8.12.0　　(最新バージョン)

sendmail は sendmail-8.10 以前と以後に分けられます。

SMTP Auth の機能を使わないであれば、sendmail-8.9.3 で十分です。また日本語版の sendmail.cf 作成ツール WIDE CF(CF-3.7Wpl2)が対応しているので便利です。基本的に sendmail のバージョンアップはセキュリティホールのよるバージョンアップではなく、機能拡張がメインとなります。SMTP Auth 等の機能を使用したい場合には、むやみに最新バージョンを使用しない方がよいでしょう。

機能を比較すると、sendmail-8.12.0 は sendmail-8.11.6 からメジャーバージョンアップばかりであるということがわかります。メジャーバージョンアップしたアプリケーションのインストールは十分に注意する必要があります。理由はバグが潜んでいる可能性が高いからです。sendmail-8.12.0 のみが持つ機能の必要がなければ、sendmail-8.11.6 をインストールすべきです。

4.4 余談（バージョンアップ）

新しいバージョンが出たからといって、むやみにバージョンアップを行ってはいけません。特に必要がなければ、バージョンアップを行なわないのが基本です。理由は、

① バージョンアップによって不安定になる可能性がある
② バージョンアップによるバグがある可能性がある
③ サービスを停止しなければいけない

からです。バージョンアップの際は、別の環境を用意して検証を行い、問題ないことを確認してから本番機で行なうべきです。

4.5 mc

mc とは sendmail 付属の sendmail.cf 作成ツールです。マニュアルがよくないので、基本的に使用方法が難しくなっています。あらゆるバージョンに付属しているので、sendmail.cf を作ることができなくなるということはありません。OS ごとに設定内容が異なり、OS ごとに対応させる必要があります。サンプルが付属していますのでそれを使用すれば基本的な部分は足ります。

使用方法

mc を OS 環境に合わせて作成後、それを利用し sendmail.cf を作成します。

例

BSD 系 OS	generic-bsd4.4.mc
Solaris2	generic-solaris2.mc
Linux 系 OS	generic-linux.mc

サンプル

（Solaris の場合、基本的に OS 依存の部分以外は他の OS でも使用可能）

```
divert(0)dnl
VERSIONID(`$Id: generic-solaris2.8.mc,v 8.11 2001/09/11 gshapiro Exp $')
OSTYPE(solaris8)dnl
Dwnakoruru
Dmico-g.net
DOMAIN(generic)dnl
define(`MAIL_SETTINGS_DIR', `/etc/mail/')dnl
define(`DATABASE_MAP_TYPE', `hash')dnl
define(`confDONT_BLAME_SENDMAIL', `GroupWritableDirPathSafe')dnl
define(`confTRUSETED_USERS', `majordomo slist')dnl
define(`confPLIVACY_FLAGS', `goaway,noetrn')dnl
define(`confTO_QUEUEWARN', `1d')dnl
define(`DATEBASE_MAP_TYPE', `hash')dnl
define(`confCOPY_ERRORS_TO', `postmaster')dnl
FEATURE(`access_db')dnl
FEATURE(`mailertable')dnl
```

```
FEATURE(`virtusertable')dnl
FEATURE(`genericstable')dnl
FEATURE(`nouucp', `reject')dnl
FEATURE(`use_cw_file')dnl
MAILER(local)dnl
MAILER(smtp)dnl
```

CF

CF とは WIDE で作成された sendmail.cf 作成ツールです。

利点

① 日本語マニュアルが付属しているので、mc に比べて楽に sendmail.cf の作成が可能です。
② マニュアルがしっかりしています。

問題点

① mc ほど細かい設定はできませんが実用的に問題はないでしょう。
② 対応が sendmail-8.9.3 までであり、Sendmail-8.10.0 以降でも使用できますが、Sendmail-8.10.0 すべての機能が使えるわけではありません。
③ CF で作成した sendmail.cf と mc で作成した sendmail.cf では書き方が全く異なるので、手で書き換えなければいけない場合には十分に注意する必要があります。

4.6 Security（SPAM）

Mail 環境のセキュリティ問題

SPAM が主問題となるでしょう。

SPAM とは？

SMTP のリレー配信機能を悪用して、悪意ある他人が配信したメイルを大量に配信させられること。大量メイル配信に利用されているメイルサーバを踏み台サーバとよばれます。

SPAM の問題

踏み台にされた場合には大量のメイルを送付するので、サーバの負荷は増加します。その結果、限界を超えた場合、サーバが停止し、サービスに影響を与えることになります。一種の Dos 攻撃です。

踏み台にされることはその管理者が SPAM 対策設定を行なわなかったことが原因です。踏み台として使われた方が悪いというのが現状です。しかも踏み台にされた結果、他のメイルサーバに影響を与えることになります。どちらかといえば、こちらのほうが重要問題です。

対策

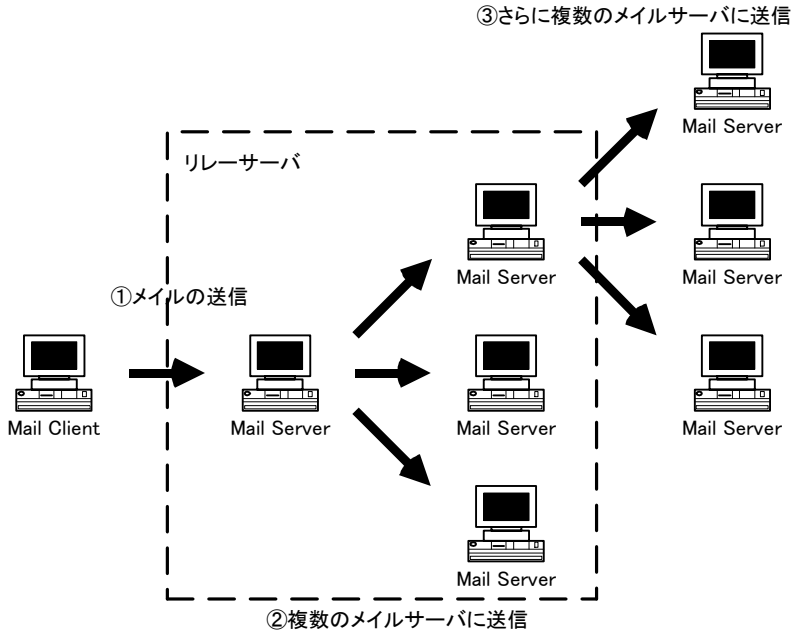
① リレーをされないように設定します。現状ではデフォルトでリレーしない設定になっているので、故意にリレー設定を行なわないかぎり問題ありません。
② ただ単に、リレーを行なわない設定を行った場合、ローカルにあるメイルクライアントからのメイル送信までもリレーしようとしているとみなしはじいてしまうので、ローカルのメイルに関してはリレーを許可するようにします。

4.7 SMAP の仕組み

SMAP の仕組みは鼠算と同じです。あるメイルサーバにおくると、そのメイルサーバが複数のアドレスに対してメイルを送信し始めます。それが連鎖的に発生します。

基本的に SPAM は SMTP のリレー機能を利用します。SMTP では end to end のメイルサーバで通信を行なうのが一般ですが、なんらかの理由（例えば、メイルサーバが Private Segment にあるとか）により、任意のメイルサーバを経由して送信します。この場合、中継に利用されるメイルサーバはリレーの機能を利用してメイルの転送を行ないます。

注意しなければいけないのは、Mail Client がメイルを送信する場合、SMTP のプロトコルを使用してメイルサーバと通信を行ないます。Mail Client は MX を用いてメイルの送信をするわけではないので、設定されたメイルサーバに代わりにメイルを送信するように依頼します。この場合、送られたメイルサーバは、「リレーを依頼された」と認識します。



4.8 SPAM 対策（mc）

mc の場合

① config.mc の中身に以下の記述を追加します。

```
define(`MAIL_SETTINGS_DIR', `/etc/mail/')dnl
define(`DATABASE_MAP_TYPE', `hash')dnl
FEATURE(`access_db')dnl
```

② /etc/mail/access を作成します。

例えば、192.168.1.0/24 からのメイルをリレーさせるためには

```
192.168.1 RELAY
```

とします。これで、192.168.1.0/24 からのメイルはリレーされます。それ以外はリレーされません。

③ データベースを作成します。

```
# makemap hash /etc/mail/access < /etc/mail/access
```

とすると、/etc/mail/access_db が作成されています。

4.9 SPAM 対策（CF）

CF の場合

① sendmail.def の設定

```
# [smtpcheck]
#MAIL_RELAY_RESTRICTION=yes                        ←default
#WITH_OLD_CF=no # (just for smtpcheck.def)
##CHECK_HOST_ALLOW=/etc/sendmail.allow
##CHECK_HOST_DENY=/etc/sendmail.deny
#CHECK_RELAY_DEFAULT=allow # (allow/deny)
# LOCAL_HOST_* does not check senders address
LOCAL_HOST_IPADDR=/etc/sendmail.localip          ←変更
                          （ファイルに IP Address のリストを書いて制御する場合）
LOCAL_HOST_IPADDR=130.54.0                       ←変更
                          （sendmail.cf に IP Address を書いて制御する場合）
LOCAL_HOST_DOMAIN=/etc/sendmail.localdomain      ←変更
                                （ファイルにドメイン名リストを書いて制御する場合）
LOCAL_HOST_DOMAIN=sub.kyoto-u.ac.jp              ←変更
                                （sendmail.ｃｆにドメイン名を書いて制御する場合）
```

ドメイン名で制御する場合は「From」行を参照するのが基本です。この行は騙ることができるので、使用すべきではないでしょう。IP Address で制御すべきです。

② ファイルにリストを記載するのならば「 /etc/sendmail.localip」、
「 /etc/sendmail.localdomain」を作成します。

/etc/sendmail.localip

```
192.168.0
10.109.
```

/etc/sendmail.localdomain

```
ico-g.com
ico-g.ne.jp
```

注意

IP Address での制御の場合、Classless の対応はできません。

4.10 Qpop

Qpop

ユーザがメイルを読むためだけに, メイルサーバにログインすることは希です。多くのユーザは, PC 上で動作するメーラ (MUA) を利用しています。これらのメーラは、サーバに telnet などでログインせずとも、 POP3 や IMAP を利用してメイルサーバからメイルを PC にダウンロードします。

popper とは POP (Post Office Protocol) サーバのことです。POP サーバには幾つかの種類がありますが、インターネット上では Qpop というサーバプログラムが広く利用されています。世界で最も広く使われている SMTP サービスプログラムである sendmail との整合性が良く、設定が簡単で安心して使えるプログラムです。

Apop

APOP は従来の POP3 サーバ・クライアントにおけるユーザ認証システムについてセキュリティ面を強化したものです。

POP サーバに接続する場合、ユーザ認証を行なうためにそれぞれのアカウント及びパスワードを用いますが、通常の POP の場合、サーバ･クライアント間で行われるユーザ認証の通信が平文で行われるため、セキュリティ的に危険でした。APOP では、このパスワードをやり取りする際、暗号化して送るため、今までの POP を用いるよりもセキュリティの向上がはかれます。

また、パスワードの管理方法が異なり、APOP を利用する場合はサーバ自体のパスワードと POP でのパスワードは全く別個の物となるため、サーバ運用の観点からも有益です。

APOP はより安全にメールパスワードの処理を行なうもので、メイル(通信)内容については暗号化されるわけではありません。メイル(通信)内容は、通常通り暗号化されずに送信されます。

4.11 Qpop の構築

構築前に

Qpop は基本的に初期状態で使用しても問題はありません。Qpop の初期環境設定は「configure」を用いて行ないます。

注意すべきところとしては、

① 基本的に Sendmail を使用する際に使用するもの（qmail には付属している）。
② inetd 経由で使用するのか、standalone モードで使用するかを考える。どちらを使用するかは好み。inetd 経由で使用する場合、tcp_wrapper が使用できる。
③ セキュリティ的に問題があるので、あまりインターネットからアクセスできるような場所に Qpop をインストールしたサーバを設置すべきではない。可能ならば Apop を使用すること。
④ 「configure」のオプションで、「--enable-debugging」のオプションがあるが、log が増大するので、あまり使用すべきではない。
⑤ Log の収集方法には２種類がある。Syslogd を使用するものと、独自のファイルに収容するものである。デフォルトでは syslog を使用する。独自ファイルの場合は起動時のオプションで対応する。
⑥ Syslog の場合、デフォルトとファシリティは「local0」である。変更は「configure」で行なう。例えば、local7 に変更する場合は「--enable-log-facility=LOG_LOCAL7」である。
⑦ 基本的にユーザアカウントは必要となる。パスワードは別のものも使用できる。

ということです。

構築

構築方法は下記のとおりである。

```
> ./configure
> make
> su
# make install
```

4.12 Qpop の設定

inetd 経由の場合

/etc/inetd.conf への追加

```
pop3 stream tcp nowait root /usr/local/libexec/popper popper
                                          ↑
                                  popper のある場所の絶対パス
```

Standalone の場合

```
/usr/local/sbin/popper –S
                          ↑
                Standalone のオプション
```

適当な Script を書いて所定のところに置いておくと（FreeBSD ならば「/usr/local/etc/rc.d」、Solaris ならば「/etc/rc2.d」）、起動時に自動的に立ち上がります。

例　Solaris の場合（/etc/rc2.d/S99popper）

```
#!/sbin/sh
case "$1" in
'start')
if [ -f /usr/local/sbin/popper ]; then
echo 'sshd service starting.'
/usr/local/sbin/popper -S
fi
;;
'stop')
/usr/bin/pkill -x -u 0 popper
;;
*)
echo "Usage: $0 { start | stop }"
exit 1
;;
esac
```

4.13 Apop の構築

構築前に

Apop の基本は Qpop で使用するパスワードを暗号化しただけのものであり、メイルそのものを暗号化するものではありません。Qpop のソースの中に Apop は含まれており、構築時のオプション設定で構築を行ないます。暗号化仕様の差異がいくつか存在しています。

① MD5 を使用するもの（正確にいえば暗号化ではないの）
② Kerberos5 を使用するもの
③ Openssl を使用するもの。これは単純に暗号化するわけではなく、通信パケットまで暗号化を行ないます。使用する場合、構築の際の指定以外に CA 局の認証を行なえるようにしなければいけません。ただし、あくまで Apop Server と Mail Client の間を暗号化しているだけで、MTA 間の通信を暗号化はしていません。

他に、

① 基本的に、gdbm が必要です。
② パスワードは専用のものを使用します。このため、ユーザを作成しても、telnet 等を使用して login できないように設定が可能です。
③ パスワードが暗号化されており、かつシステムにリモートでログインできるパスワードを使用しないので、インターネットのアクセスに対して Security は高くなっています。
④ その一方で、対応している Mail Client が少ないです。

を考える必要があります。

構築

```
> ./configure –enanle-apop=/etc/pop.auth
> make
> su
# make install
```

Apop でも作成されるコマンドは「popper」です。基本的な設定方法は「popper」と同じです。

4.14 Popper のセキュリティ

Popper

インターネットの中で核になるメイルサービスを担うサービスなので、どこでも使用しているサービスです。しかし、Qpop ではセキュリティ上の問題からいくつかの制限を受けています。基本的なセキュリティ問題とは、暗号化されないパスワードがインターネット上を流れるということです。

① インターネットから直接アクセスできるところには設置しない。
② ISP ではインターネットから直接アクセスできないようにしている。
③ そのため、あまりインターネットからの使い勝手がいいものとはいえない。

対策

いろいろな対策が考えられています。

① セキュリティ上問題があるとわかっていても対策を特に何も取れないので対策をなにもしない。
② インターネットからのアクセスを行なう場合、ルータで 110 番のポートのみをオープンする（実際には他もオープンすることになる。HTTP、DNS 等）
③ ダイアルアップサーバを内部に設置し、そこからのアクセスを許可する。
④ パスワードをこまめに変更する（実際問題困難）
⑤ APOP を使用する（Mail Client を選ぶので、注意が必要）

4.15 imap4

Imap4 とは？

imap4 はメイルを読むためのサービスのひとつです。メイルを読むシステムとしては popper と同じですが、基本的なシステムの考え方が異なります。

popper　Mail 本体を Mail Client に保持する。基本的にサーバ側にメイルは残らない。
imap4　Mail 本体をサーバに保存するため、Client の環境に依存しない。Imap4 に対応している Mail Client であるならば、どこからでも利用が可能である。

Imap4 と popper の利点と欠点

Imap4 と popper の利点

popper

① Mail 本体が利用者の Mail Client に転送されるため、Mail Spool を大量に用意する必要ありません。
② 通信費はメイル受信時のみのため、接続時間が短くすみます。

Imap4

① Client 側にメイルを Spool しないので、利用者のメイルを管理しなければいけない場合、一括して管理ができます。
② 必要なメイルだけを読むことができるので、大容量のメイルが添付されており、それを受信する必要がない場合、転送しないですむので、通信データ量（トラフィック）を少なくすることができます。
③ メイルを受信する必要がないので、PDA の様な記憶容量の少ないメディアでも利用が可能です。

Imap4 と popper の欠点

Popper

① Client に Spool された Mail の管理は原則ユーザー任せとなります。事実上管理者による管理は不可能です。ISP の場合、基本的に利用者の Client に spool された Mail は管理する必要はあります。

② PDA といった記憶容量の少ないメディアでは、受信容量等の制限で使用できない場合があります。

Imap4

① サーバに大量のメイルをスプールするための領域が必要となります。

② 大量にアクセスがあった場合、アクセスが重たくなる可能性があります。

③ あまりいい imap4 サーバが存在しません。機能のサービス可能・サービス不可能はサーバに依存します。フリーでは「cyrus」もしく「imap-uw」程度です。

4.16 imap4 の利用場面

imap4 と popper の利用の現状

現状では、popper が主に利用されています。これは、ISP では管理コストの問題がメインとなるからです。ユーザー数が少人数であれば問題も少ないのですが、多人数になった場合、必要となる記憶容量は巨大になるので対応できるメディアが少なくなります。例えば、一人あたり 20MB のスプールを用意した場合、10000 ユーザを収容するならば単純計算で 200GB を必要です。200GB を使用するのが現実的かどうかがという問題もありますが、200GB の HDD システムを用意し維持することは困難です。また障害時の対応も popper の方が Mail Spool のバックアップのみで対応出来るので、管理コストは下がります（imap4 サーバの場合も、Mail Spool のバックアップは必要）。

imap4 の利用場面

① PDA 等の記憶容量が少ないメディアを利用する場合
② 利用料金が接続時間ではなく、パケット数で課金される場合
③ 利用者のメイルを管理しないといけない場合
④ 同じ端末を使用しないが、同じメイルを閲覧したい場合

4.17 imap-uw の構築

構築

```
# make neb        (NetBSD/FreeBSD)
```

```
# make gso        (Gcc Solaris)
```

Install

```
> su
# cp ipopd/ipop2d /usr/local/libexec
# cp ipopd/ipop3d /usr/locla/libexec
# cp ipopd/imapd /usr/local/libexec
```

参考

① ipop2d は pop2、ipop3d は pop3 用のプログラムなので imapd 的には必要ありません。

② 「/usr/local/libexec」というディレクトリがない場合には作成してください。必ずこのディレクトリでなければいけないという理由はありませんが、「/usr/local」の下に適当なディレクトリを作成してください。

l/etc/inetd.conf の設定

下記の行を追加します。（FreeBSD はコメント行になっているので、コメントをはずします）

```
imap4 stream tcp nowait root/usr/local/libexec/imapd imapd
                              ↑
                インストールした imapd の絶対パス
```

4.18 Imap-uw のセキュリティ

セキュリティ

基本的に通信を暗号化するものはありません。

① 認証方式は「typically Password 認証」・「kerberos 認証」・「Cram-MD5 認証」が使用できます。他に「PAM 認証」も存りますが、Mail Client が多くありません。ちなみに MD5 は Hash 関数であり、定義上では暗号ではない。

② セキュリティ的には少し弱い。

設定例　基本編　Mail Server で POP Server もしくは APOP Server を兼用

適用ケース　OCN　ADSL（固定 IP Address 付与サービス）

利点

① 機器台数が少ない

② 低コスト

問題点

① POP Server の場合、セキュリティ的な問題が存在します。

基本的な運用の方向性

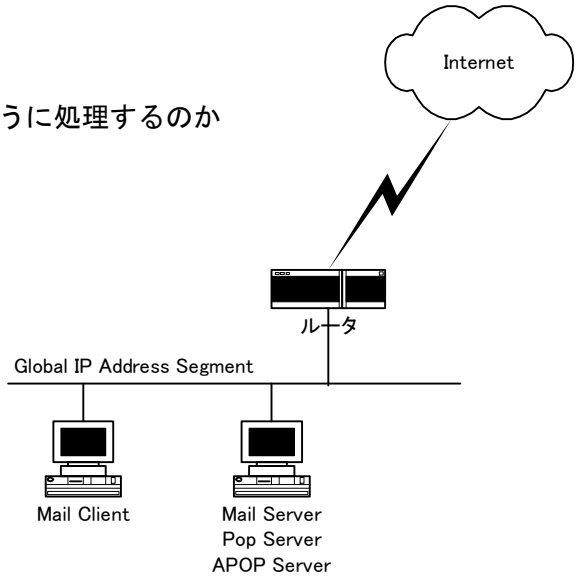
② インターネットからのアクセスをどのように処理するのか

(ア) 特に何もしない

(イ) ルータによるアクセスフィルタ

(ウ) サーバでのアクセスフィルタ

(エ) APOP を使用する



sendmail.cf の設定

① 特に特別な設定はなし

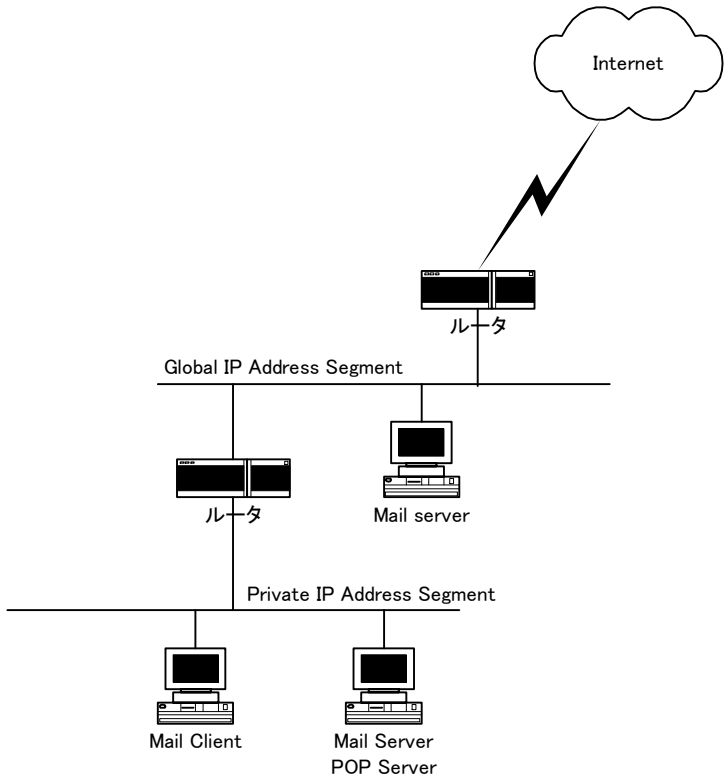
② 同一セグメントからのリレーを許可

設定例　　POP Server を Private IP Address Segment に設置する方法

利点

① POP Server が Private IP Address Segment にあるため、Internet からのアクセスが不可能であるため、セキュアです。

② Global IP Address Segment に設置した Mail Server にすべての利用者のアカウントを作成する必要がないので、Global 側の管理が楽である。



欠点

① インターネット側から POP Server へのアクセスが不可能であるため、Dial Up Server 等のシステムを Private IP Addres Segment に設置するなどして、外部からアクセスできるようにしなければいけません。

② 機器台数が多くなります。

③ 機器コストが高くなります。

設定ポリシー

① Global 側は internet 側からのメイルはすべて Private 側の Mail Server にメイルを自動的に送信します。

② Private 側の Mail Server はインターネット側に Mail を送信する場合 Global 側の Mail Server を投げるように設定します。

③ Private 側の Mail Server からインターネットに投げられた Mail は、必ず Global 側の Mail Server から出されたように変更をかける必要があります。これは、エラーメールが帰ってこなくなる可能性がるからです。

設定例　　FireWall 形状のメイルシステム 1

基本ネットワーク

① ail Server を FireWall 的に使用する方法です。

② ail Server で NATD、Proxy Server を動作させれば、インターネットへのコネクティビティは確保できます。

利点

① Router 等を用意しないので、安価に構築が可能です。

② Private Segment を作るので、この部分のセキュリティを低く設定できるので融通が利きます。

問題点

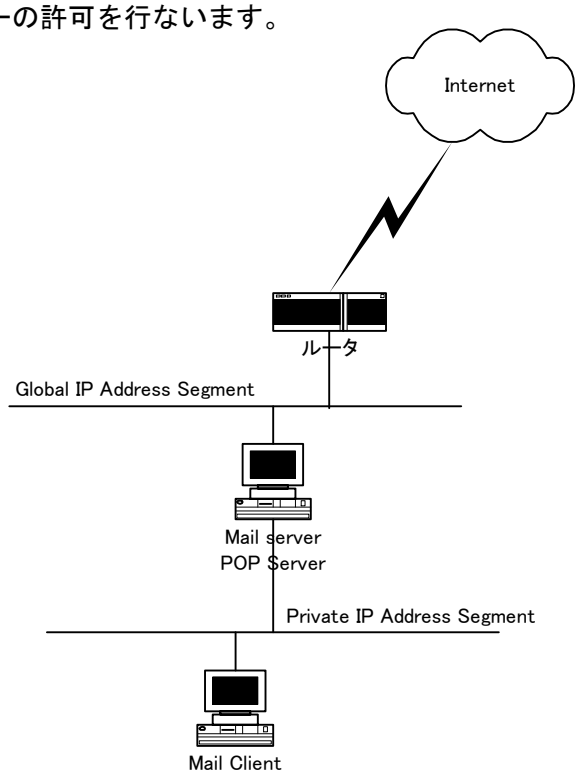
① Mail Server がクラックされた場合、Private Segment は無防備になります。クラックされないように、細心の注意が必要です。

② インターネットからの POP へのアクセスが可能であるが、セキュリティ的な問題があります。可能ならば APOP を使用するべきです。

メイルの基本設定

① 特にありません。

② Private Segment からのリレーの許可を行ないます。



設定例　　FireWall 形状のメイルシステム 2

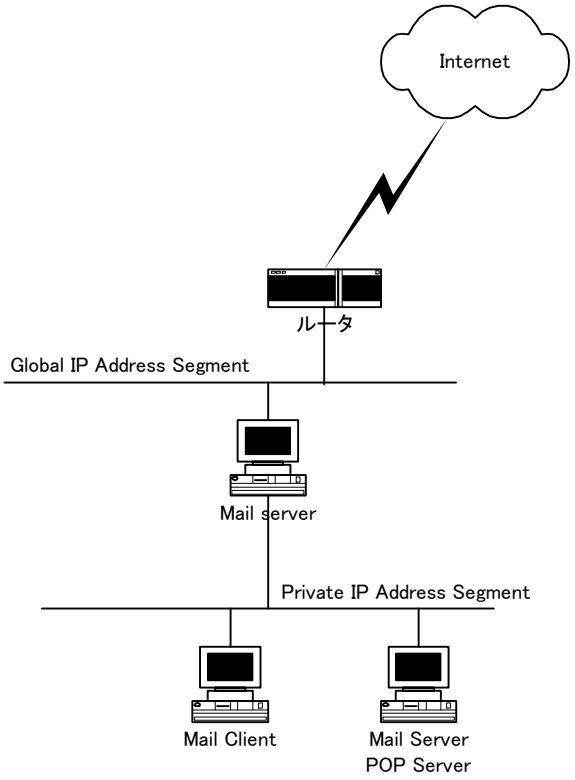
基本ネットワーク

① メイルスプールを Private Segment に設置。これにより、インターネットからの POP への不正アクセスを防ぐことができます。

② Global Segment にある Mail Server は最低限の機能だけで十分です。機能分散が図れます。

メイル環境設定

今までの混合方式です。



4.19 POP Before SMTP

SMTP で認証が必要な理由

本来、SMTP では自由に利用されるべきものですが、SPAM 等のサーバに悪影響を及ぼすようなことを行なう輩がいるために利用の制限を行なわなければならないのです。

ローカルからの使用であれば使用される IP Address は既知であるため、IP Address でフィルタできますが、インターネット側からではどの IP Address から利用するのかが不明になります。インターネットからのリレーを不許可にすることは可能であり、いちばん簡単な方法です。しかし複数の ISP と契約しており、利用内容によって接続する ISP が異なる場合、毎回 SMTP の設定を変更する必要があります。この方法は管理負担が増加し、可能であるのならば同じ SMTP サーバをたほうが管理負担は低くなります。しかし各 ISP では個人に割り当てられる IP Address は動的に割り当てられるので、SMTP 側ではどの IP Address から利用されるのか指定できません。そこで、IP Address で SPAM の防止を行なうのではなく、認証をおこなって認証に成功した利用者に対して、リレーの許可を行なう方法が有効になります。

SPAM 防止方法の種類

SMTP の SPAM を防止する方法としては、

① IP Address、hostname 等を利用してアクセスフィルタを作成する方法
　(ア) 利用する IP Address だけを許可する方法
　(イ) SPAM 元になっている IP Address を不許可にする方法

② メイル送信の際に認証を行なう方法
　(ア) SMTP AUTH
　　SMTP に認証機能を加え、メイルを送る際に認証を行い。認証に成功した利用者に対してメイルのリレーを許可する機能。Mail Client の対応が十分でない。
　(イ) POP before SMTP
　　SMTP AUTH と違い、SMTP で認証を行なうのではなく、POP での認証に成功した利用者に対してメイルのリレーを許可する機能。基本的に認証に POP を使用するので Mail Client に依存しないため、現在のところ主流の方法。

があります。

POP before SMTP の仕組み

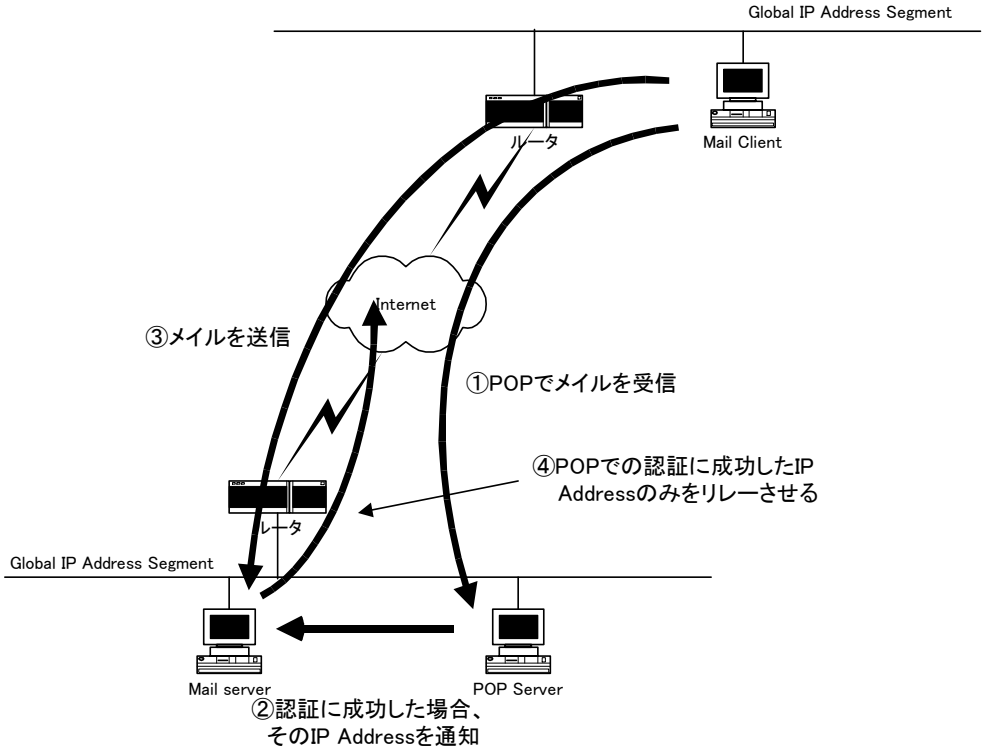
（ア） メイルの送信を行なう前にメイルの受信をおこないます。この際に POP での認証を試行します。

（イ） POP の認証が成功すると、POP 認証に成功した Mail Client の IP Address を Mail Server に通知します。

（ウ） Mail Server は知らされた IP Address をデータベースに追加します。

（エ） Mail Client はメイルを送信します。

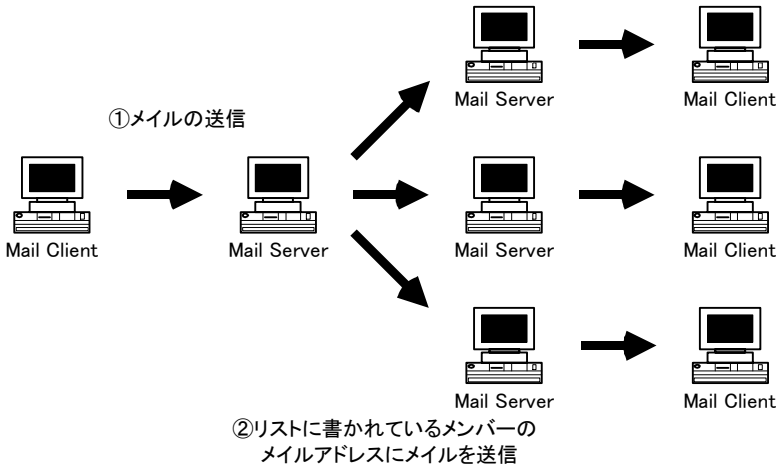
（オ） Mail Server は Mail Client の IP Address とデータベースと比較し、マッチした場合メイルをリレーします。マッチしないならばリレーを行いません。



4.20 Mailing List(ML)

Mailing List とは？

一通のメイルをあるメイルアドレスに対して送ると、そのメイルを複数宛てに送るメイルシステムです。宛先はリスト化しており、そのリストに書かれているアドレスに対してメイルを送ります。ちなみに ML システムがメイルを出しなおすことになるのでリレーにはなりません。



4.21 Mailing List の構築方法

Mailing List の構築方法には、①Mail の alias を使用する方法と②Mailing List 用のアプリケーションを使用する方法があります。

Alias による Mailing List 作成

/etc/aliases にエントリを追加します。

メンバーが少ない場合

```
#hogehoge Mailing List
owner-hogehoge: hogehoge-request
hogehoge-request: mailing-list-owner@mail.ico-g.com
hogehoge: uja@ico-g.net, hoge@ico-g.com
```

Mailing List Owner のアドレス

メンバーのアドレスを記載

メンバーが多い場合

```
#hogehoge Mailing List
owner-hogehoge: hogehoge-request
hogehoge-request: mailing-list-owner@mail.ico-g.com
hogehoge: ":include:/var/spool/mailing_list/hogehoge_ml
```

Mailing List Owner のアドレス

メンバーのアドレスを記載

/var/spool/mailing_list/hogehoge.ml の中身

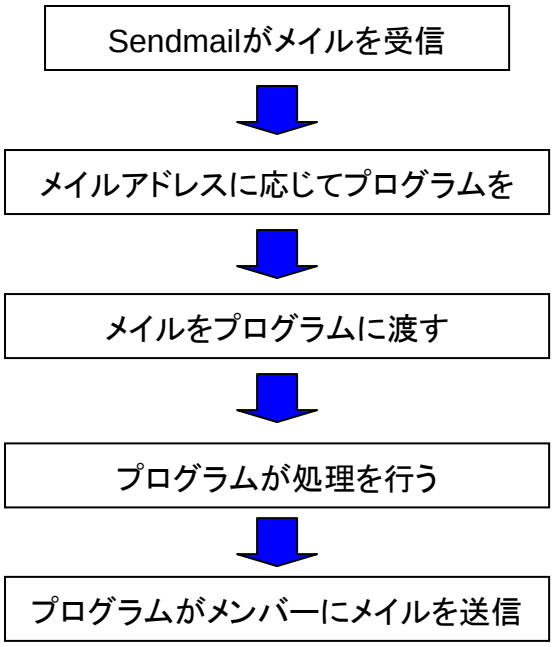
```
uja@ico-g.net
hoge@ico-g.com
ujauja@aaaa.com
```

Alias による Mailing List とは

- 基本的に送られてきたメイルを多人数にフォワードするものです。
- 簡単に作れ、どこからでもだれからでもメイルを送ることができます。制御を行ないたい場合には、sendmail 側で行ないます。
- メイルの内容の変更（ヘッダの追加、シーケンス番号の追加等）はできません。
- 送られてきたメイルのアーカイブはできません。

4.22 Majordomo による Mailing List

基本的なシステム



処理そのものは sendmail が行なうわけではないので、多種多様な対応が可能です。

① 送信メイルのアーカイブ
② メイルヘッダーの追加
③ アクセス制御
④ 過去のアーカイブの閲覧
等

Sendmail とは別のプログラムが起動するので、セキュリティに注意します。最悪の場合、セキュリティホールになる可能性があります。

5章　HTTP 環境構築

5 HTTP 環境構築

5.1 HTTP 環境サービス

サーバで対応するサービス

- ホームページの閲覧 → 基本設定
- アクセス制限 → htaccess の設定
- CGI → CGI の設定
- JAVA → サーバでは何もしない
- PROXY → PROXY の設定
- 暗号化（SSL） → SSL の設定

サーバで対応しないサービス

- JAVA

HTTP Server の種類

以下は主な HTTP Server です。

- Netscape Communication Server Netscape 社製
- Internet Information Server Microsoft 社製
- Apache FreeWare

HTTP のセキュリティ的な問題

HTTP は比較的ユーザーレベルで自由度のある設定が可能です。そこで設定内容には注意が必要です。例えば、CGI の扱いをどうするかという問題があります。ユーザに自由に自作の CGI を使えるようにすることもできるし、させないこともできます。多くの ISP では使えないようにしています。これはユーザの自作の CGI にセキュリティホールがあった場合、システムに影響を及ぼす可能性があるからです。ユーザが自作した CGI を検証してから使わせるようにする方法も対策として考えられますが、運用上現実的ではありません。管理者側で用意した CGI 以外は使わせないようにする ISP が多いのが現状でしょう。

セキュリティ的にどこまでユーザに自由度を許すかはサービス次第です。しかし、約款でしばることは簡単であるが、ユーザは故意でなくてもシステムに影響を及ぼすような行為を犯す可能性があります。どのような場合でも問題のないようなシステム構築が必要です。

5.2 Apache の構築

Apache の構築前の初期設定

```
>./configure      [オプション]
```

単純に立ち上げるのだけであるのならば、オプションを付ける必要はありません。Apache の場合、各機能はこの段階で使用するしないの設定を行ないます。細かい環境設定は httpd.conf 内で行ないます。いくつかの機能はデフォルトで設定されています。その機能を使用したくない場合、明示的に使用しない環境設定にする必要があります（実際、httpd.conf 内で競っていなければ使えませんが、使わない機能ならば可能な限り明示的に外した方が無難でしょう）。基本的にデフォルト設定に問題のある設定は特にありません。

apache の機能（モジュール）

Ⅰ　Apache-1.3.19.tar.gz の展開

```
> cd ${WORK_DIRECTORY}
> tar xvfz apache-1.3.19.tar.gz
```

Ⅱ　初期設定

```
> cd ${DWORK_DIRECTORY}/apache-1.3.19
> ./configure --with-port=80 --with-layout=Apache
```

Ⅲ　コンパイル

```
> cd ${DWORK_DIRECTORY}/apache-1.3.19
> make
```

Ⅳ　インストール

```
> su
# make install
```

Ⅴ　環境設定

必要に応じて行います。基本的に default でも使うことは可能ですが Security 的には問題あります。

Ⅵ　起動

```
# /usr/local/apache/bin/apachectl start
```

5.3 Apache の Directory 構造

Apache のデフォルト

/usr/local/apache	bin	apache の binary を収容するディレクトリ
	libexec	ライブラリの収容するディレクトリ
	icons	デフォルトのアイコンを収容するディレクトリ
	conf	コンフィギュレーションファイルの収容するディレクトリ
	logs	アクセスログ、システムログの収容するディレクトリ
	htdocs	ホームページのルートディレクトリ
	proxy	PROXY のキャッシュを収容するディレクトリ
	cgi-bin	CGI を収容するディレクトリ
	man	apache のマニュアルを収容するディレクトリ

GNU のレイアウト

/usr/local/	bin	apache の binary を収容するディレクトリ
	sbin	apache のサーバ系の binary を収容するディレクトリ
	libexec	ライブラリの収容するディレクトリ
	share/icons	デフォルトのアイコンを収容するディレクトリ
	etc	コンフィギュレーションファイルの収容するディレクトリ
	logs	アクセスログ、システムログの収容するディレクトリ
	share/htdocs	ホームページのルートディレクトリ
	share/proxy	PROXY のキャッシュを収容するディレクトリ
	share/cgi-bin	CGI を収容するディレクトリ
	man	apache のマニュアルを収容するディレクトリ

ディレクトリポリシー（apache に限ったことではないが）

ディレクトリりポリシーをインストール前に決めておくことが大切です。

1. GNU 系で統一

GNU 的にある程度決められたディレクトリにインストールします。例えば、一般のバイナリは /usr/local/bin、サーバ用バイナリは/usr/local/sbin、マニュアルは/usr/local/man に収容するといったものです。これは慣れれば非常に便利な方法です。ただし、BIND の様に使用するファイルが少ないアプリケーションは問題ありませんが、apache のように使用するファイルが多い場合にはそれがシステム内に分散されるので、初心者には管理しにくいものとなります。

2. アプリケーションのデフォルト

各アプリケーションで設定されているデフォルトのディレクトリを使用する方法です。これは初心者向けです。必要なファイルはそのディレクトリの下に入っているので管理しやすくなります。その一方でバイナリやマニュアルが分散するので、それ専用の環境設定が必要となります。

3. 折衷案

GNU 系とデフォルトの折衷案です。両者のいいところを組み合わせる方法です。

4. 個人のポリシー

単純に個人の好きにポリシーを作成します。初心者にはあまりお薦めできません。なぜならば、好みは時間がたつにつれて変化するので、インストールした場所が時期によって異なることになります。業務で使用している場合に引継ぎ時に苦労することになります。

5. では、どれがいい？

最終的には好みで決定します。データ量が増減するようなもの（ログとか）を「/usr/local」の下に入れることは良いことではないという考えもあります。慣れてしまえば、どの方法でも運用上問題はないでしょう。可能であるならば GNU 系をデフォルトで扱えるようになるべきです。これは他人が構築したシステムでもある程度理解できるからです。

ディレクトリポリシー（例）

GNU 系変型

/usr/local/	bin	apache の binary を収容するディレクトリ
	sbin	apache のサーバ系の binary を収容するディレクトリ
	libexec	ライブラリの収容するディレクト
	man	apache のマニュアルを収容するディレクトリ
/etc/apache		コンフィギュレーションファイルの収容するディレクトリ
/var/log/apache		アクセスログ、システムログの収容するディレクトリ
/var/share/apache	htdocs	ホームページのルートディレクトリ
	share/icons	デフォルトのアイコンを収容するディレクトリ
	proxy	PROXY のキャッシュを収容するディレクトリ
	cgi-bin	CGI を収容するディレクトリ

折衷型

/usr/local/	bin	apache の binary を収容するディレクトリ
	sbin	apache のサーバ系の binary を収容するディレクトリ
	libexec	ライブラリの収容するディレクトリ
	man	apache のマニュアルを収容するディレクトリ
/usr/local/apache	icons	デフォルトのアイコンを収容するディレクトリ
	conf	コンフィギュレーションファイルの収容するディレクトリ
	logs	アクセスログ、システムログの収容するディレクトリ
	htdocs	ホームページのルートディレクトリ
	proxy	PROXY のキャッシュを収容するディレクトリ
	cgi-bin	CGI を収容するディレクトリ

Apache の環境設定（httpd.conf）

Apache の環境設定（httpd.conf）

構築の際にきちんと設定をしていれば、基本的なとこは設定されています。ディレクトリ構造はここでも変更は可能です。構築の際に設定した場所にインストールされ、自動的に httpd.conf が生成されます。そのままでも使用可能です。好みで変更したほうがいいところもあるでしょう。

環境設定の注意

I. 各ディレクトリの設定〔Options の追加〕

サーバ自身の設定以外では、実際にコンテンツを置くディレクトリの設定が主です。ここで、各ディレクトリごとにできること、できないことの設定を行ないます。

① アクセス制御
② CGI の実行の可否
③ できることできないことの制御

II. 各ディレクトリでのできることできないことの制御

基本的にオプションを付けることにより許可を行ないます。ただし書き方によってはユーザから htaccess を利用して上書きも可能になるので、注意が必要です。

AllowOverride None ←上書きの禁止
AllowOverride All ←すべての操作を許可

III. Options の設定

必要に応じて設定を行ないます。必要でないならば設定は行ないません。消去すべきです。デフォルトで追加している「Indexes」は、あまりよい設定ではないので削除すべきです。また複数の設定を１つの設定で行なうものもあるので注意が必要です。

環境設定

コンパイル時の初期設定から自動生成されるところとされないところがあります。一通り設定が間違っていないかを確認し、問題がある場合は修正します。

ServerType　Server の起動方法を「inetd」から立ち上げるのか、もしくは「Standalone（単独）」で立ち上げるのかを設定します。inetd からの起動では inetd 系のオプションも使えますが、基本的に Standalone です。

ServeRoot　Server の Root Directory を設定です。Log のディレクトリに影響します。ファイル、もしくはディレクトリの場所を相対で記述した場合、ここがカレントになります（自動生成）

PidFile　Server の PID を収容するファイル名です。適当な場所を書いておきます。基本的にディレクトリは存在させておく必要はあります。ファイルは必要ありません。ディレクトリがないと「apachectl」で再起動できません。

ScoreBoardFile　適当な場所を設定します。

Port　Apache でオープンする Port 番号。デフォルトでは 8080。80 に変更する必要です。

User　サーバにアクセスしてきたユーザ名。大抵は「nobody」です。

Group　サーバにアクセスしてきたグループ名。大抵は「nobody」です。

ServerAdmin　基本的に製作者のアドレスが割り当てられます。「webmaster」等のアドレスを作成し、それを使います。基本的に alias で管理者に転送します。製作者のアドレスを書いてはいけません。

ServerName　サーバ名を記入します。DNS で引くことが出来るホスト名とは違う名前にしたい場合に使用します。例えば「www.ico-g.net」です。ちなみに、サービスに関係するホスト名を FQDN で使用すべきではありません。「www.ico-g.net」というようなサービスに使用するホスト名は CNAME で対応します。これはシステム変更の際に便利です。

DocumentRoot　サーバ名でアクセスした場合に表示されるページを収容するディレクトリです。例えば「http://www.ico-g.net」とした場合に表示されるコンテンツがある場所です。このディレクトリが存在しない場合、Server は起動しないので注意が必要です。

ディレクトリ制御

ディレクトリごとに制御を行なうのが基本です。

例えば、DocumentRoot の設定は、

```
<Directory /usr/local/apache/htdocs>
Options FollowSymLinks MultiViews
AllowOverride None
</Directory>
```

となります。つまり、<Directory /usr/local/apache/htdocs > ～　</Directory>までが　「/」のディレクトリの環境設定となります。

アクセス制御はディレクトリごとに制御を行ないます。

```
<Directory “/usr/local/apache/htdocs">
Options FollowSymLinks MultiViews
AllowOverride None
Order deny,allow              ←アクセス制御の順番。
Deny from all                 ←あらゆる場所からアクセス拒否
Allow from  .ujauja.com       ←「ujauj.com」からのアクセスを許可
Allow from 192.168.2.0/24     ←「192.168.2.0/24」からのアクセスを許可
</Directory>
```

アクセス制御の場合、「Order」行の「allow」、「deny」の順番が重要です。IP　Address でも制御が出来ます。

CGI の問題

CGI の問題

セキュリティ的な問題です。

CGI とはブラウザを経由して、誰でもプログラムを実行できる機能です。不用意に使用できるような設定を行なうとセキュリティホールとなる可能性があります。これを使用する場合には細心の注意が必要です。Office LAN 程度であれば、管理者が利用者が使用したい CGI を管理することは可能であるが、ISP 等では事実上不可能です。

対策

① CGI でできることを制限します。

使用できるコマンドの制限

② ユーザ独自の CGI は使用できないようにする。

(ア) ユーザで実行権限を設定できないようにする。

単純に「chmod」を使用できないようにする。

(イ) Apache 側で実行できないようにする。

ユーザが利用するディレクトリに

AllowOverride None

Options MultiViews SymLinksIfOwnerMatch IncludesNoExec

等（Options は適当）の設定を施すと、そのディレクトリ以下でユーザによる設定の上書きが出来なくなります。「Options」の設定で、

ExecCGI

は設定しません。これを設定すると CGI の使用が可能となります。

余談

CGI というものは基本的にサーバで CGI プログラムを実行し、その結果を HTML の形式で出力するものです。そのため、CGI プログラムはサーバで動作しなければなりません。実行権限がなければ実行できないので、当然 CGI は使えなくなります。

CGI の設定

CGI プログラム

基本的にサーバ上で実行可能なプログラムです。

スクリプト言語（sh、perl、ruby 等々）

C 言語　等々

注意

CGI というものは基本的にサーバで CGI プログラムを実行し、その結果を HTML の形式で出力するものです。そのため、CGI プログラムはサーバで動作しなければなりません。

動作方法

基本的に拡張子に依存します。

- .pl
- .cgi
- .sh

必ず CGI プログラム名の最後につける必要があります。設定は変更可能です。他の拡張子でも試用できます。

```
AddHandler cgi-script cgi pl sh
```

共用 CGI の環境設定(デフォルトで設定されている)

```
<IfModule mod_alias.c>
  ScriptAlias /cgi-bin/ "/usr/local/apache/cgi-bin/"   ←URL 上でのエイリアスの設定
  <Directory "/usr/local/apache/cgi-bin">              ←CGI 用のディレクトリの設定開始
      AllowOverride None                               ←念のため
      Options None                                     ←ExecCGI でなくても OK
      Order allow,deny                                 ←アクセス制御
      Allow from all                                   ←あらゆる場所からのアクセス可
  </Directory>                                         ←CGI 用のディレクトリの設定終了
</IfModule>
```

Apache-SSL

ブラウザと Web サーバ(httpd)との通信は、Hyper Text Transfer Protcol (HTTP) と呼ばれるプロトコルでデータをやり取りします。しかし、生データが流れるためインターネットショッピング等をする際に、個人情報が漏れるなどセキュリティ上不具合が生じます。そこで、データを暗号化する Secure Sockets Layer (SSL)という仕組みが考えられました。

SSL そのものは Netscape Communications 社が開発したインターネット上で情報を暗号化して送受信するプロトコルです。SSL は公開鍵暗号や秘密鍵暗号・デジタル証明書・ハッシュ関数などのセキュリティ技術を組み合わせ、データの盗聴や改ざん・なりすましを防ぐことができます。OSI 参照モデルではトランスポート層(第 4 層)にあたり、HTTP や FTP などの上位のプロトコルを利用するアプリケーションソフトからは特に意識することなく透過的に利用することができます。

Apache-SSL は Apache と SSLeay をベースにしたセキュリティのあるウェブサーバです。最近のバージョンは、SSLeay でなく、OpenSSL を使うようになっています。Apache-SSL は BSD スタイルのライセンス に基づいており、著作権掲示を維持する限り、商用・非商用に限らず自由に利用できます。これは、Apache バージョン 0.8.15 以降と同じライセンスの内容です。

特徴

① 商用、非商用共に無償で利用可能
② 世界的な 128 ビット暗号機能
③ クライアント認証機能
④ 全ソースコードを公開
⑤ 拡張モジュラーAPI

Access　Log の取り扱い

Log の形式は httpd.conf で設定します。

LogLevel warn　←　log Level。警告を log に残す設定。あらゆる接続情報を Log に残す場合は、「debug」にするといい。ただし、情報量は多いので注意が必要。

LogFormat “%h %l %u %t \”%r\“ %>s %b \”%{Referer}i\“ \”%{User-Agent}i\“” combined　←　Log Format

LogFormat “%h %l %u %t \”%r\“ %>s %b” common　←　Log Format

LogFormat “%{Referer}i -> %U” referer　←　Log Format

LogFormat “%{User-agent}i” agent　←　Log Format

CustomLog logs/access_log common　←　Log の書き込み先とフォーマットを指定。

#CustomLog logs/referer_log referrer　←　Log の書き込み先とフォーマットを指定。コメント

#CustomLog logs/agent_log agent　←　Log の書き込み先とフォーマットを指定。コメント

#CustomLog logs/access_log combined　←　Log の書き込み先とフォーマットを指定。コメント

推奨設定

Loglebel	debug
Customlog	logs/access_log combined

理由

この設定ではあらゆるアクセス情報が Log として収容されます。これを解析すれば、どこからアクセスがあるのか、アクセスが多い時間帯、どのブラウザからアクセスしているのかといった情報がわかります。

参考(log 解析ツール）

analog

5.4 VirtualHost

VirtualHost とは？

１つの HTTP Server で複数のドメインのホームページを立ち上げることです。利用面としては ASP、ISP での独自ドメイン サービスがあります。これらのサービスでは利用者のドメインのサーバを立ち上げています。これらのサービスでは１ドメインで１つのサーバを使用しているのではなく、複数のドメインで１つのサーバを使用しています。複数のドメインを１台のサーバで管理することにより、機器スペースと管理の両方のコストを低減しています。

方法

1.ドメインごとに IP Address を割り振り、アクセスを切り替える方法
2.Hostname だけでアクセスを切り替える方法

1 の問題点は、Virtual Host ごとに IP Address を必要なので、多数の IP Address を必要とします。2 では必要ありません。2 の問題点では、ブラウザによっては正確に表示されないブラウザもあります（かなり少ない）ので、そういうブラウザの場合には対応できません。１の場合はそのようなことはありません。可能であるならば、１で行ないましょう。ISP では基本的に１で行っていますが、APNIC では２を推奨しています。しかし、サービス的に２ではトラブルが生じる可能性がある以上、薦められません。

構築の際の考え方

Apache では、設定上で Virtual Host でないものと、Virtual Host であるものとを同時に立ち上げることが出来ます。ブラウザでアクセスしている限り差違はありませんが、管理上では全く異なります。そのため、すべてを Virtual Host 扱いをした方が管理上便利ですが、Apache では Virtual Host でないものの立ち上げが必須となっています。いわゆる DocumentRoot を設定しなければいけません（Apache 以外では Virtual Host しかないのもある）。管理上では、DocumentRoot では独自ドメインの設定は行なわず、すべて Virtuahost 側で設定するほうが便利でしょう。例えば

Hostname	uja.hoge.com	←	DocumentRoot の設定
Virtualhost	www.hoge.com	←	VirtualHost の設定
	www.ujauja.com	←	VirtualHost の設定

とし、www.hoge.com を hoge.com の URL とすれば、www.hoge.com にアクセスすることになります。１つしかドメインを必要としないのならば、Virtualhost を設定する必要は全くありません。

VirtualHost の初期設定

構築

VirtualHost はデフォルトの構築では設定されていません。構築の初期設定時にモジュールを追加します。

```
> ./configure --enable-module=vhost_alias
> make
> su
# make install
```

注意

もし以前から Apache を使用していた場合、環境設定ファイル（httpd.conf）はバックアップを取っておきましょう。

IP Address による Virtualhost の設定

IP Address による Virtualhost の設定

```
<VirtualHost 192.168.2.5>
      ServerAdmin webmaster@hoge.com
      DocumentRoot /usr/local/apache/virtualhost/hoge/htdocs/
      ServerName www.hoge.com
      ErrorLog logs/www.hoge..com-error_log
      CustomLog logs/www.hoge.com-access_log combined
</VirtualHost>
<VirtualHost 192.168.2.6>
      ServerAdmin webmaster@ujauja.com
      DocumentRoot /usr/local/apache/virtualhost/ujauja/htdocs/
      ServerName www.ujauja.com
      ErrorLog logs/www.ujauja.com-error_log
      CustomLog logs/www.ujauja.com-access_log combined
</VirtualHost>
```

注意事項

VirtualHost に使用するアドレスは（使用する IP Address に対して Ping が通る等）サーバに対してネットワーク的に使用可能な IP Address を使用します。１つのネットワークインターフェイスに複数の IP Address を設定しマルチホームコンピュータを構成します。

FreeBSD の場合の例(/etc/rc.conf に下記エントリを追加)

ifconfig_de0_alias0 ="inet 192.168.2.5 netmask 255.255.255.255"

ifconfig_de0_alias1 ="inet 192.168.2.6 netmask 255.255.255.255"

DNS の設定（エントリの登録・各 resolv の設定）をしなければ、ホスト名でのアクセスできなくなります。（IP Address でアクセスは可能です。）

HostName による VirtualHost の設定

```
NameVirtualHost www.hoge.com
NameVirtualHost www.ujauja.com      <VirtualHost www.hoge.com>
ServerAdmin webmaster@hoge.com      ←あまり個人のメイルアドレスにすべきではない
DocumentRoot /usr/local/apache/virtualhost/hoge/htdocs/
ServerName www.hoge.com
ErrorLog logs/www.hoge..com-error_log     ←適当な場所。VirtualHost ごとにログを
                                          収集
CustomLog logs/www.hoge.com-access_log combined ←適当な場所。VirtualHost ごと
                                                にログを収集
</VirtualHost>
      <VirtualHost www.ujauja.com>
      ServerAdmin webmaster@ujauja.com    ←あまり個人のメイルアドレスにすべきではな
                                          い
      DocumentRoot /usr/local/apache/virtualhost/ujauja/htdocs/
      ServerName www.ujauja.com
      ErrorLog logs/www.ujauja.com-error_log    ←適当な場所。VirtualHost ごと
                                                にログを収集
      CustomLog logs/www.ujauja.com-access_log combined     ←適当な場所。
                                                VirtualHost ごとにログを収集
</VirtualHost>
```

注意

DNS へのエントリ登録は必須となります。設定しなければアクセスできません。（IP Addres では判断できません。）

CNAME で設定します。

5.5 ネットワーク構成例

Proxy

Proxy とは？

いわゆる代理サーバとか踏み台サーバといわれているサーバです。全てではありませんが、ブラウザで使用する機能をブラウザの代理でアクセスし、結果をブラウザに転送する機能を持ちます。基本はキャッシュにアクセスしたコンテンツを収容します。もしブラウザの要求に対してコンテンツがキャッシュにある場合にはキャッシュからコンテンツが配信されます。これの利点は下記のとおりです。

① コンテンツがキャッシュにある場合、インターネットにアクセスしないので、コンテンツのブラウザへの転送が早くなる。
② コンテンツをインターネットにアクセスする回数が減るので、対外回線の容量負荷が低減される。
③ Proxy を経由すれば、Private IP Address からのブラウザのアクセスが可能となる。

Proxy 製品

- Soft Ware
  - ① Apache　　Web Server に付属していますがあまり使われてはいません。理由は、サーバはサーバマシン１台で重たいサービスを１つ提供する場合が多いことから、Web Server として使うサーバと Proxy として使うサーバの２台用意する構成を取ることになり、このような場合わざわざ Apache で Proxy Sever を構築せず別のアプリケーションで構築します。あまり使い勝手がいいものではありません。
  - ② Squid Proxy 専用の Application(FreeWare)では一般的です。

  等々

- Hard Ware
  - ① Cash Server　　箱物なので高価

  等々

Apache での Proxy

構築

Proxy はデフォルトの構築では設定されていません。構築の初期設定のときにモジュールを追加します。

```
> ./configure --enable-module=proxy
> make
> su
# make install
```

注意

Proxy では使用する Port 番号を web server と異なるものを使用します。（慣習として 8080 や 10080 を使用することが多いでしょう。）Apache では Web Server と併用する場合に、port 番号は web server と同じになり分離が出来ません。結局 Apache が Proxy として使用されないのは、ここに理由があります。

Apache の Proxy の設定

設定

```
<IfModule mod_proxy.c>
ProxyRequests On  ←Proxy の動作を on
<Directory proxy:*>     ←アクセス制御。インターネット側から無条件でアクセスさせるわ
                        けにはいかないので制御する。
      Order deny,allow
      Deny from all
      Allow from .ujauja.com  ←必要なアドレスだけを許可する
</Directory>
      ProxyVia On
      CacheRoot “/var/spool/apache/proxy“ ←キャッシュの場所。適当なところを用意し
                                          ておく必要がある
      CacheSize 5000          ←キャッシュとして 5000KB を使用する
      CacheGcInterval 4       ←キャッシュをチェックする時間の感覚（４時間）
      CacheMaxExpire 24       ←キャッシュされたコンテンツを保持する時間（24 時間）
      CacheLastModifiedFactor 0.1
      CacheDefaultExpire 1
</IfModule>
```

テクニック

ここでは詳細は記載しませんが、インターネット側からプライベートセグメントのコンテンツにアクセスする際にも Proxy の環境を使用することができます。

6章　FTP 環境構築

6 FTP 環境構築

6.1 FTP の基本機能

FTP とは遠隔地にあるコンピュータとの間でファイルのやり取りをする手順ファイル転送プロトコル(File Transfer Protocol)のことで、 このプロトコルに対応したサーバを FTP サーバといいます。

電子メイルでもファイルのやり取りは出来ますが大きいファイルになると、 転送が遅くなったりメールストアの容量制限により送れなくなるので、大きいファイルの場合は FTP の利用が便利です。

FTP 機能を利用するには、以前は専用の FTP クライアントソフトを使うのが一般的でした。しかし最近は、ブラウザで FTP サイトにもアクセスできるしデータの転送もできます。

Anonymous FTP

通常は、相手のコンピュータに接続するための ID やパスワードが必要となります。しかし、たくさんの利用者が自由に情報入手を行なえるよう、パスワードの必要なしにファイルのやりとりができる FTP サーバも存在します。ここでは多くのフリーウェア・シェアウェアが登録されており、利用者は匿名でこのサーバにアクセスし、必要なファイルやプログラムを入手することができます。このような Internet 上で一般に開放されている FTP サーバは、 anonymousFTP（アノニマスエフティーピー）というものです。

anonymous アノニマスとは匿名の事です。AnonymousFTP では匿名で FTP（ファイル転送）ができます。

anonymousFTP を利用するときは、

アカウント（ユーザ名） → anonymous

パスワード → 自分のメイルアドレス

を入れるのが一般的です。

6.2 Anonymous FTP Server

I. Wu-ftpd-2.5.1 の構築

```
> ./build CC=gcc bsf    (ただし、FreeBSD の場合。)
> su
# ./build install
```

II. インストールされるもの（default）

/usr/local/libexec/in.ftpd

/usr/local/bin/ftpshut

/usr/local/bin/ftprestart

/usr/local/bin/ftpcount

/usr/local/bin/ftpwho

/usr/local/bin/privatepw

Anonymous FTP Server の設定

I. ユーザーの登録

ftp というユーザを登録し、/etc/passwd のファイルを開いて編集します。（すでに登録されているなら下記の記述と同じものかどうかを調べます。）

```
ftp:*14:50:FTP User:/home/ftp:
```

ユーザ ID とグループ ID は、他と重ならなければなんでもかまいません。

パスワードは*で潰しておきます。

また、ログインシェルも本物のシェルではなく、適当なダミーを置きます。

II. グループの作成

/etc/group に ftp という名前のグループを作ります。**dir** コマンド実行時の表示の問題と、グループが重複して割り当てられないように確保する意味があるので作成しておくとよいでしょう。

III. anonymous FTP 用ディレクトリ作成

anonymous FTP 用のホームディレクトリとそのサブディレクトリを作成します。

Anonymous FTP のユーザはユーザ名 ftp として root アクセス権が与えられるので、システム関係のファイルが置かれるディレクトリは所有者を root に 755 モードをにしておきます。ftp ユーザの権限で書き込みを可能にはしません。

（例）

① **mkdir** で ftp ディレクトリを作成します。
② ディレクトリの所有者を **chown** で root に、モードを **chmod** で 755 にします。
③ 作成した ftp ディレクトリの中に etc と bin ディレクトリを作成します。
④ ディレクトリの所有者を root に、モード 111 を 755 にします。

IV. passwd ファイルの作成

先ほど作成した etc ディレクトリの中に passwd ファイルを作成します。このディレクトリには ftp の下にファイルを置く可能性のあるユーザだけを登録します。このファイルに登録されていなければ所有者やグループの表示が数字になるだけの事なので安全性からいうと root と ftp だけにしておくのがよいでしょう。また、このファイルのパスワードは必ず潰しておきます。決して本物の /etc/passwd をコピーしてそのまま使わないように！

（group も同様に作成する）

（例）

/home/ftp/etc/passwd ファイル

```
root:*:0:0:root:/root:/bin/bash
ftp:*:14:50:FTP User:/home/ftp:
```

/home/ftp/etc/group ファイル

```
root::0:root
ftp::50:
```

V. FTP 用バイナリの用意

anonymous FTP からアクセスできるように/home/ftp の下に bin ディレクトリを作成してバイナリを置きます。必要なライブラリは、**ldd /bin/ls** コマンドで知ることが出来ます。それらを /home/ftp/bin にコピーします。さらにも/usr/lib/ld.so を/home/ftp/bin にコピーし、/home/ftp/bin のモードを 555 に設定します。

VI. 検証

ftp localhost を実行し、ログインを試みます。

7章　サーバ管理

7 サーバ管理

7.1 サーバの設置場所

基本的にサーバは普段コンソール作業を行なうものではありません。特にサーバの機能とは関係のない作業を行なうべきではありません。必要なときだけログインし必要な作業のみを行なうようにします。一般にサーバはネットワークシステムの中で比較的中心となる存在です。不用意に作業を行なうことで障害を発生させた場合、その規模の大小に関わらず、利用者に何らかの影響を及ぼします。（サービスを提供する意でサーバと呼ばれる事からも）サーバを設置するということはサービスを提供すると同意であり、利用者に影響を及ぼすような管理を決してしてはいけません。サーバを運用管理者の身近な場所（机の下等）に置いておくことも出来ますが、何らかの作業中に電源ケーブルを抜いてしまう・電源ボタンを押してしまう等のトラブルが起きないとも限りません。

実際には設置場所としては、

① 不用意に触ることが出来ないようなところ（ラック等）

② 電源、温度等が管理され、防火施設が整っているているところ（データセンタ等）

に設置します。

注意

実際問題として、完全に安全な場所におけるかといえば予算等の問題で不可能である場合が多いでしょうが、可能な限り安全な場所に設置すべきです。

7.2 サーバの管理

サーバの日々の管理

サーバは基本的に手元には設置しません。サーバルームのラックに設置したり（ラックが机の横に置いてあるようなところは稀でしょう）、データセンタに設置したりします。実際問題として、１～２台程度ならば管理者の手元においてあるかもしれないが、ISP では多数のサーバを使用するので、全てのサーバを管理者の手元に置いておくことは事実上不可能です。だからといって、必要なときだけサーバの設置場所まで行き作業をすることは非効率です。まして、データセンタであるならばそこまで行くのに時間がかかる可能性があります。そこで通常はリモートで作業を行い、ハードウェア交換などサーバの設置場所での作業が必要な時にのみ、サーバの設置場所で作業を行ないます。

リモートでの作業方法

管理していく過程で root になる場合があります。この時にセキュリティに注意します。最も重要なのはパスワードです。暗号化をしていない状態で root password のリモート使用をしてはいけません。必ず SSH 等で暗号化をします。絶対に **telnet** を使用してはいけません。また、SSH を使用する際も無秩序にインターネットからアクセスさせるのも問題です。暗号化されているとはいえアクセス方法は **telnet** と同じなので、アタックそのものの防御としては **telnet** と大差ありません。必ずアクセスできる IP Address を限定すべきです。

リモート作業の決まりごと

·SSH を使用する事。可能な限り RSA 認証・DSA 認証・ワンタイムパスワードも併用します。
·SSH を使用する場合、アクセス制御を必ずすること。

注意

これらを使用すると、利用する際の使い勝手が悪くなる。使い勝手が悪くなるから使わないというような管理者は管理者として不適格なので決して管理者にしてはいけない。

サーバの日々の作業

サーバを管理する日々の作業として、下記の点に注意を払いましょう。

1. プロセスに異常はないか？
    ① 普段よりプロセスが増減していないか？
    ② 立ち上がっているべきプロセスがきちんと立ち上がっているか（立ち上がっていても、止まっているのがある）？
    ③ みたことがないプロセスがあがっていないか？
    ④ みたことがないプロセスが使用されていないか？
2. HDD の利用状況
    ① 利用率が高くないか？
3. CPU 負荷
    ① CPU 利用率は高くないか？
4. Syslog に異常を示すメッセージは書き込まれていないか？
    ① Syslog にみたことのないメッセージが書き込まれていないか？
5. アクセス
    ① アタックされていないか？

7.3 Syslog の管理

システム管理をする上で必要となるあらゆる情報記録のことを一般にシステムログまたはシスログと呼びます。ログを収集することで①システムの利用傾向の把握と②潜在的な問題の発見が図れます。これらの情報を収集することで大きな障害が発生する前に対処をすることも出来ます。システムもしくはアプリケーションで障害があった場合にこれを解析し障害原因を解明できます。普段から管理するシステムのログをとりその傾向をつかんでおくことが大切です。

システム上、アプリケーションの障害・警告等の情報は一般に syslog を収集するプログラム（syslogd）を通して収集されます。収集場所は、

/var/log/**** BSD 系 OS（FreeBSD 等）
/var/adm/**** SVR4 系 OS（Solaris 等）

等です。

設定は/etc/syslog.conf で行ないます。

7.4 Process 管理

UNIX は多数のプログラムが同時に起動しているマルチタスク OS です。起動しているアプリケーションは Process ID(PID)で管理されています。同じプログラムは複数起動することもあるので、その起動ごとにも PID で管理されています。また、起動しているアプリケーションにはそれぞれ所有者（起動者）が存在します。UNIX はコンソール以外からでも接続が可能なため、同じ所有者でもどの terminal から接続されているかによっても管理制御されます。

PID の確認

ps [オプション]

オプション	意味
x	ターミナルを使用しないプロセスの表示
a	他のユーザのプロセスも表示

オプションの詳細は OS によって異なリます。なにもオプションを付けない場合は、その利用者の権限で起動しているアプリケーションのみが結果として表示されます

Process 管理とセキュリティ

UNIX の管理する上で不必要なアプリケーションを立ち上げるべきではありません。特にサーバとして機能するアプリケーションは必要ない場合には立ち上げません。不必要なサーバの起動は管理されていないサーバと同意です。アプリケーションの持つセキュリティの脆弱性を理解して使用しているならば問題ありませんが、仕様上セキュリティホールでなくても管理していないアプリケーションがセキュリティホールになる場合があります。このようなことを防ぐために、不必要なサーバ機能を有しているアプリケーションは立ち上げるべきではありません。

Process の停止

プロセスの停止方法

kill [オプション] [*PID*]

オプション	意味
1	HUP (hang up)
2	INT (interrupt)
3	QUIT (quit)
6	ABRT (abort)
9	KILL (non-catchable, non-ignorable kill)
14	ALRM (alarm clock)
15	TERM (software termination signal)

通常はほとんどオプション 1(HUP)と 9(KILL)のみの使用となります。

オプションは数字・アルファベットの両方が使用できます。1(HUP)は起動しているアプリケーションを停止し、再起動するためのものです。9（KILL）は別の terminal から起動したアプリケーションを停止します。

注意
オプションを使用する際に、アルファベットを使用するようにします。数字を使うと「kill –1 119」となりますが、間違えて「-」を付け忘れた場合「kill 1 119」となり、PID が 1 と 119 のアプリケーションを停止することとなります。PID1 は init です。init が止まるとシステムも停止します。

7.5 データの保護

データの保護はサーバ管理上重要であす。これは、

① 障害時のサーバの早期復旧のため
② 利用者のデータの保護

等の理由があります。基本的にサーバは必ず故障するものとして考えます。故障しないサーバは存在しません。しかし、サービスを行っている以上、サービスを停めるようなことは出来る限りないようにします。サービスを停止する場合にはできる限り短時間で復旧させるべきです。

データの保護の基本的な方法としては、

① バックアップ
② RAID1（ミラーリング）
③ RAID5（冗長構成で、ホットスタンバイ可）

等があります。RAID１・RAID5 は即時対応ですが、バックアップは 1 日に 1 回・1 週間に１回といった周期で行なうので、リアルタイムに対応できません。障害から復旧する場合、最新のバックアップをとったところまでしかデータは存在しません。基本的に RAID5 に対応していればデータは保護されますが、バックアップもとっておいたほうがより安全です。RAID にはハードウェア RAID とソフトウェア RAID がありますが、ハードウェア RAID の使用を推奨します。ハードウェア RAID は OS に依存しません。

8章　参考　RAID

8 参考 RAID

RAID や SAN (Storage Area Network) など専用のハードウェア ソリューションを実装すると、CPU の負荷が軽減されます。ただし、これはハードウェア ソリューションに独自の処理機能がある場合だけです。

1 台の大容量ディスクを使用するよりも、小容量のディスクを複数使うほうが、パフォーマンスは向上します。たとえば、36 GB のデータ格納が必要な場合でも、1 台の 36 GB のディスクを使用するのでなく、4 台の 9 GB のディスクを使用することをお奨めします。アレイの種類にもよりますが、こうすると情報の書き込みが最大で 4 倍に高められる例もあります。

8.1 RAID テクノロジ

RAID にはいくつか種類がありますが、どのタイプの RAID も、複数の物理ディスクにデータを分散し、アプリケーションからデータを格納するロジックとは、まったく異なるロジックでデータを格納します。

本書では、RAID-0、RAID-1 (およびサブタイプ 0+1)、RAID-5 という 3 種類の RAID と 1 つのサブタイプの実装方法について説明します。これ以外にも多くのタイプの RAID がありますが、現在使用されている RAID ソリューションのほとんどは、この 3 種類です。

RAID-0

RAID-0 は複数のディスクに並列に読み書きするストラプを行なうもので、この「ストライプ」により、複数のディスクに保管したデータを 1 つの論理パーティションと見なされます。このようにして、複数のディスクにまたがったいくつかの論理パーティションが作成できます。たとえば、実際のファイルを RAID-0 アレイに保存するときに、アプリケーションで D ドライブにファイルを保存すると設定されている場合、ファイルは複数の RAID-0 ディスク アレイに分散して書き込まれます。この例では、ファイルは 6 台のディスクすべてに分散されます。

図 1: RAID-0 ディスク アレイ

RAID-0 は一度に 6 台のディスクに書き込むため、パフォーマンスからいえば RAID の中でも最も効率的な方式です。すべてのディスクにアプリケーション データが格納される場合、ディスクの使用効率は最大になります。

RAID-0 の問題点は信頼性にやや不安があることです。Exchange メールボックスのデータベースが、RAID-0 アレイに格納されている場合、アレイ内のどれかのディスクに障害が発生した場合、障害のないディスク アレイにメールボックス データベースを復元し、トランザクション ログ ファイルを復元しなければなりません。アレイ上にトランザクション ログ ファイルを格納し、あるディスクが使用不能になった状態では、復元できるのは最終的なバックアップ時点でのメールボックス データベースだけです。

RAID-1 (および 0+1)

RAID-1 はディスク アレイ間でミラー化をするもので、2 台のディスクでミラー化を行ないます。3 台以上のディスクが使用される場合、ディスクはミラー化され、さらにストライプ化されます。これが RAID-0+1 構成です。つまり、各物理ディスクがアレイ内で重複します。6 台のディスクで構成される RAID-0+1 アレイの場合は、3 台のディスクをデータ格納に使われ、これらのディスクではストライプ化が行われ、別の 3 台のハードディスクがストライプをしているディスクを “ミラー化” します。あるディスクにデータが書き込まれるたびに、同じデータがミラー化されたディスクにも書き込まれます。

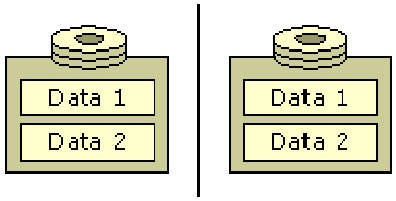


図 2: RAID-1 ディスク アレイ

RAID-1 (または 0+1) では、すべてのデータは書き込み直後にミラーされるので、信頼性の面からは、3 種類の RAID アレイの中でも最も高い方式ですが、ディスクの記憶容量効率は半分になってしまいます。不経済のように思われますが、データの格納域に高い信頼性が必要な場合には、RAID 1 (または 0+1) をお薦めします。

RAID-5

RAID-5 は RAID-0 と同様に、ストライピングによりデータをアレイ内の各ディスクに分散して保存します。ただし、RAID-5 にはパリティ機能があります。つまり、アレイ内のあるディスクに障害が発生した場合、残りのディスクからデータが再構築できるように、アレイに格納されたデータの整合性を保持するためのメカニズムが備えられています。RAID-5 も信頼性の高いストレージ ソリューションです。

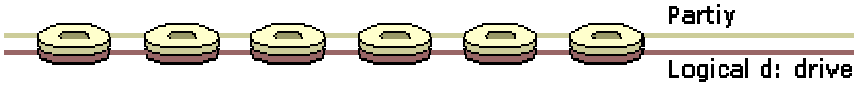


図 3: RAID-5 ディスク アレイ

しかしパリティ用に、各ディスクに 1/n GB のディスク容量が必要になります。n はアレイ内のドライブの数です。たとえば、9 GB のディスクが 6 台ある場合 (合計 54 GB)、使用可能な記憶容量は 45 GB になります。パリティを保持するために、1 回のデータ書き込みごとに、RAID-5 アレイでは 2 回の書き込みと 2 回の読み込みが実行されます。このため、全体的なパフォーマンスは低下します。

RAID-5 ソリューションの長所は、高い信頼性を持ちながら、RAID-1 (および 1+0) よりも効率的にディスクのリソースが使用できることです。

8.2 RAID ソリューションの比較

ストレージ サイズは、異なる使用環境でもほぼ一定であるため、あるストレージ サイズを基準にして、これらの RAID ソリューションのコスト・パフォーマンス・信頼性が比較評価できます。表 1 は以下の条件を前提としたものです。

- 90 GB のデータを格納する。
- 9GB のドライブを使用する。
- アレイは、1 秒あたり 100 回という入出力 (I/O) 速度でディスクにデータを書き込む。

表 1: RAID ソリューションの比較

RAID ソリューション	ドライブの数 (コスト)	最大書き込み回数/秒	最大読み取り回数/秒	信頼性
RAID-0	10	1000	1000	低い
RAID-0+1	20	1000	2000	最大
RAID-5	11	275	1100	高い

RAID-1 ソリューションではディスクを 2 台しか実装できないため、表に RAID-1 は含まれていません。RAID-1 では、90 GB のデータを格納するために、45 GB ドライブが 2 台必要になりますが、スループットは大幅に低下します。

信頼性は、ディスクの障害時にデータがどのくらい正確に保持されるかを示すものです。RAID-0 は冗長性を持たないため、RAID-0 アレイでディスクの障害が 1 つでも発生するとデータを完全に復元しなおさなければならなくなります。RAID-0+1 では 2 つ以上のディスクの障害が発生するまではデータは失われず、特定のディスク セットに障害が起きないかぎりデータは失われないため、最も信頼性の高いソリューションです。

コストの評価は、アレイを構成するために必要なディスク数から計算されます。RAID-0+1 では、実際にデータを保存するために必要なディスク容量の 2 倍のディスク容量が必要で、コストは最も高くなりますが、最大読み取りおよび書き込み回数に見られるように、RAID-5 に比較してパフォーマンスはかなり高くなります。