日本大学歯学部紀要 33，83-102，2005

インターネット・サーバーの構築

宮崎　洋一[1]　磯川　桂太郎[2,6]　網干　博文[3,7]
江島　堅一郎[4,8]　大山　哲生[5,9]　山崎　洋介[2,6]

Setting up the Internet server

Yoichi Miyazaki[1], Keitaro Isokawa[2,6], Hirofumi Aboshi[3,7], Ken-ichiro Ejima[4,8], Tetsuo Ohyama[5,9] and Yosuke Yamazaki[2,6]

1. はじめに

インターネット・サーバーを管理・運用できるようになるためには、どのくらいの勉強が必要になるだろうか。サーバーを3婆ー、Webで利用されるCGIを4爺eyeと書いても違和感がない(?)ように、かなりの老練さ（＝学習）が要求されることは想像に難くない。

一口にサーバーといっても、そこではメールサーバーやWWWサーバーを始めとする様々なプログラム（デーモン）が動作しており、それぞれのサービスを提供している。したがって、サーバーの管理・運用のためには、サーバーのOSであるUnixはもちろんのこと、各サービスごとの仕組みについて深い知識と技術が必須である。

著者の一人は、2003年11月にサーバーのマシンの入れ替えを機に、OSのバージョンアップとサービス・プログラムの更新を行なった。このとき、OSにはPlamo Linux3.3(Kernel 2.4.22-v01)を使用し、D. J. Bernstein氏によるプログラム群、いわゆるdjbツールを中心に据えサーバーの設定を行なった。古いマシンからのデータ移行やiptables, clockspeedなどの新規導入のため、設定作業には連休を含め一週間ほどかかった。

この解説では、小規模なネットワークにおけるサーバーを想定し、その構築例を紹介する。それは、このときのインストール記録を元にしているが、将来新たにサーバーを構築するとしたら、こんな感じに設定したいという計画書でもある。執筆にあたっては、その後バージョンアップされたPlamo Linuxのもとで部分的に動作確認を行なっている。

サーバーをいわゆるdjbツールによって固めたのには、いくつか理由がある。第1の理由は、djbツールはセキュリティ・ホールなどの

日本大学歯学部　[1]数理情報学教室，[2]解剖学教室第2講座，
[3]法医学教室，[4]放射線学教室，[5]補綴学教室局部床義歯学講座
日本大学歯学部総合歯学研究所　[6]機能形態部門，
[7]社会歯科学部門，[8]高度先端医療研究部門，
[9]顎口腔機能研究部門
〒101-8310　東京都千代田区神田駿河台1-8-13
（受理：2005年11月7日）

[1]Department of Applied Mathematics and Informatics, [2]Department of Anatomy, [3]Department of Legal Medicine, [4]Department of Radiology, [5]Department of Removable Partial Denture Prosthodontics, Nihon University School of Dentistry
[6]Division of Functional Morphology, [7]Division of Social Dentistry, [8]Division of Advanced Dental Treatment, [9]Division of Oral and Craniomaxillofacial Research, Dental Research Center, Nihon University School of Dentistry
1-8-13 Kanda-Surugadai, Chiyoda-ku, Tokyo 101-8310, Japan

バグがないという絶対的な信頼である。サービスのなかでも基幹となるDNS、メールの各サービスは、それぞれBIND、sendmailが標準的に用いられてきたが、しばしばセキュリティ・ホールが発見され、その度にバージョンアップが必要であった。それに比べdjbツールの対応するプログラムdjbdns、qmailには現在までセキュリティ・ホールが見つかっていない。

qmailへの移行を決定付けたのは、日経Linuxの記事[1]であった。

> 長年に渡り、世界中のシステム管理者がゴルディオスの結び目であるsendmailを切断するための剣を待ち望んできました…そして、遂にsendmailを切断する剣 — qmail — が現れた。もはやsendmailを使い続ける理由は何もない。

qmailへの移行をきっかけにその他のdjbツール群を知るようになり、徐々にそれらに移行していった。

その過程で、TEX関連の有名なプログラムYaTeXの作者でもある広瀬雄二氏の記事[2]は大変参考になった。この記事ではdjbツールを貫くプログラム哲学について、プログラマーとシステム管理者の立場から興味深い考察がなされている。あの広瀬氏が絶賛してサーバーに採用し、多くの解説記事[3, 4, 5]で啓蒙しているほどだから、というのがdjbツールを中心に据えた第2の理由である。

そして、第3の理由はあまり科学的な根拠がない、かなり単純なものである。

Bernstein!

なんといい響きだろう。英語読みでバーンシュタインと発音すれば、世界的な指揮者を連想させる。また、ドイツ語読みでベルンシュタインと発音すれば、数学における集合論の定理を思い出させる。「2つの集合AとBについて、AからBへの単射が存在し、かつ、BからAへの単射が存在するならば、AからBへの全単射が存在する」ベルンシュタインの定理と呼ばれるこの定理が、無限集合の個数を数える基本となっているように、djbツールはサーバー管理の基本となるプログラムとして、今後ますます脚光を浴びていくことだろう。

2. サーバーの構成

この節では、想定するサーバーの構成を説明する。

OSはPlamo Linux 4.0(kernel 2.4.26)とし、「おまかせパッケージ — 普通に使うには困らない環境をインストール」を選択して、Plamo Linuxがインストールされている。

サーバー管理者は、ユーザー名hiroで登録され、root権限をもっている。

以下の説明では、%で始まる行は、サーバー端末でユーザーhiroとして実行するコマンドを示し、#で始まる行は、rootとして実行するコマンドを表す。コマンドが標準出力を返すときは、コマンドの次の行にそれを示す場合もある。

```
% whomai
hiro
# whoami
root
```

なお、ここで使用するIPアドレスやドメイン名などはすべて架空のものである。

2.1. ネットワーク環境

サーバーは、某大学某学部のネットワーク内に某研究室用として設置されている。

サーバーは、2つのネットワーク・インターフェイスを持ち、表2.1のIPアドレスが割り

当てられ、eth0 側はインターネットに、eth1 側はプライベート・ネットワークに接続されている。プライベート・ネットワークには、何台かのクライアント PC が接続され、サーバーから dhcp により IP アドレスが自動的に割り振られる。

表 2.1

eth0	1.2.3.55	インターネットへ
eth1	192.168.1.1	プライベートへ

学部ドメイン nusd.jp の DNS サーバーからサブドメイン gen.nusd.jp の権限がこのサーバー (IP アドレス 1.2.3.55) に委譲されている。そして、ホスト名とその別名は表 2.2 のように設定する。

表 2.2

学部のドメイン	nusd.jp
管理するドメイン	gen.nusd.jp
ホスト名	mc.gen.nusd.jp
別名	www.gen.nusd.jp

外部と送受信するメールは、学部のサーバーによりウィルスチェックを受ける。そのために、表 2.3 のようなメール送受信の設定を行なう。

表 2.3

受信するメール ドメイン gen.nusd.jp 宛のメール ホスト mc.gen.nusd.jp 宛のメール
外部へのメール 学部のメール・サーバー mail.nusd.jp (= 1.2.3.7) に送る

クラスレスの逆引きの設定が、上位の学部ドメインのネームサーバーで表 2.4 のようになされている。

表 2.4

gen.3.2.1.in-addr.arpa ゾーンの NS 1.2.3.55
55.3.2.1.in-addr.arpa の CNAME 55.gen.3.2.1.in-addr.arpa

2.2. サービスの種類

サーバーで行なうサービスをそのプログラムのソース名とともに挙げる。

1. daemontools
   daemontools-0.76.tar.gz
   デーモンの管理とログの収集を行なう。
2. tcpserver
   ucspi-tcp-0.88.tar.gz
   ネットワーク・デーモンの監視とアクセス制御を行なう。
3. DNS　djbdns
   djbdns-1.05.tar.gz
   IP アドレスとホスト名の解決や管理を行なう。
4. SMTP　qmail
   qmail-1.03.tar.gz
   メールの送受信を行なう。
5. POP3
   imap-2002-qmav.4.tar.gz
   メール・クライアントからのメール取り込み要求に応える。
6. WWW
   (1) publicfile
   publicfile-0.52.tar.gz
   WWW サーバーとしてインターネットに公開する。トップページの URL
   http://www.gen.nusd.jp/
   (2) apache
   apache_1.3.33.tar.gz
   プライベート・ネットワークからのみ利用可能な WWW サーバー。ポートは 8080 番を使う。
   トップページの URL
   http://www.gen.nusd.jp:8080/
7. FTP　vsftpd
   vsftpd-2.0.3.tar.gz
   ファイルの送受信を行なう。

8. openssh
   Plamo Linux に標準装備
   外部からのログイン (slogin) やファイルの転送 (scp) などに応える。
9. 時刻合わせ　clockspeed
   `clockspeed-0.62.tar.gz`
   サーバーの時刻調整を行なう。
10. iptables
    Plamo Linux に標準装備
    IP マスカレードやパケット・フィルタリングを行なう。
11. その他
    (1) ML　メーリングリスト
    `CML-4.0-Release.tar.gz`
    `CML-4.0-Release_qmail.patch`
    (2) dhcpd　IP アドレスの自動割当
    Plamo Linux に標準装備
    (3) PHP
    `php-4.3.11.tar.gz`
    apache などの WWW サーバーに組み込んで使用するスクリプト言語
    (4) lpd
    プリント・サーバー

2.3. ディレクトリ構成

サーバーの設定で関連のあるディレクトリやファイルの一覧を以下に示す。

`/usr` 読込み専用のプログラム
  `/usr/sbin`　sshd
  `/usr/local/bin`　djbdns, tcpserver のプログラム群, および `ipop3d`
  `/usr/daemontools`
  `/usr/clockspeed`
  `/usr/apache`
  `/usr/vsftp`
`/service`　svscan のデーモン管理用
  `/var/`*Service* から `/service/`*Service* へリンクが張られている
`/var/qmail`　qmail の本体
  `/var/qmail/qmail-send`　svscan 管理用
  `/var/qmail/qmail-smtp`　svscan 管理用
`/var/`*Service*　*Service* という名のサービス (デーモン) の起動スクリプト `run` や tcpserver によるアクセス制御ファイル *Service*`.cdb` を置く
`/etc/inittab`　svscan の起動設定
`/etc/rc.d/rc.local`
  qmail などの起動スクリプト
`/etc/rc.d/rc.inet2`
  sshd 起動の標準設定がされている
`/home/hiro`
  ユーザー hiro のホーム・ディレクトリ
  `/home/hiro/src`
    プログラムのソースを格納する
  `/home/hiro/wtmp`
    ソースの展開と make を行なう
  `/home/hiro/wtmp/patches`
    djb プログラムのパッチを展開する

3. daemontools と tcpserver のインストール

djb のプログラムの中でも、daemontools と tcpserver は、デーモンの管理を行なうプログラムとして最も重要な位置を占めている。この節では、これらのインストール方法を説明する。

3.1. パッチの準備

djb ツールのプログラム・ソースは Bernstein 氏の Web ページ
`http://cr.yp.to/djb.html`
から入手して `/home/hiro/src` に置く。djb ツールのソースの make では、gcc のバージョ

ンが 2.3.x 以降だとエラーが出て compile ができない。Plamo Linux4.0 の gcc のバージョンは 3.3.2 なので、これに対処するためのパッチ djb_errno_patches.tgz を次の URL から入手しておく。

http://www.thedjbway.org/errno.html

◇ 動作環境などの表示

```
% uname -a
Linux mc 2.4.26-plamo #2 Sun Jun 20
14:33:31 JST 2004 i686 unknown
% gcc -v
gcc version 3.3.2
% whoami
hiro
% pwd
/home/hiro
```

◇ パッチ・ファイルの展開

```
% mkdir -p wtmp/patches
% cd wtmp/patches
% tar xfvz ~/src/djb_errno_patches.
tgz
```

この操作で次のファイルなどが展開される。

```
clockspeed-0.62.errno.patch
daemontools-0.76.errno.patch
djbdns-1.05.errno.patch
publicfile-0.52.errno.patch
qmail-1.03.errno.patch
qmail-1.03.qmail_local.patch
ucspi-tcp-0.88.a_record.patch
ucspi-tcp-0.88.errno.patch
```

◇ パッチのあて方の例

```
% cd ~/wtmp
% tar xfvz ../src/clockspeed-0.62.ta
r.gz
% cd clockspeed-0.62
% patch -p1 < ../patches/clockspeed-
0.62.errno.patch
```

3.2. daemontools のインストール

daemontools はデフォルトでは/command にインストールされるが、ここではインストール先を/usr/daemontools に変更する。そのためには少し細工が必要である。

◇ 一般ユーザーでの make と install

```
% cd ~/wtmp
% tar xfvz ../src/daemontools-0.76.t
ar.gz
% cd admin/daemontools-0.76
% patch -p1 < ~/wtmp/patches/daemont
ools-0.76.errno.patch
patching file 'src/error.h'
% ./package install
```

一般ユーザーで実行しているので、/command ディレクトリが作成できなかったというエラーメッセージが出るが、気にせず次に進む。

◇ root になって手動インストール

```
% su
# mkdir -p /usr/daemontools
# cp command/* /usr/daemontools
```

以上で daemontools のプログラム svscan、supervise、svc、svstat などが /usr/daemontools にインストールされる。

◇ サーバー起動時の svscan の設定

```
# vi /usr/daemontools/svscanboot
```

svscanboot ファイルの/command という文字列を/usr/daemontools に変更する (2ヵ所)。

```
# vi /etc/inittab
```

ファイル inittab に次の行を書き足す。

```
SV:123456:respawn:/daemon/svscanboot
```

◇ ディレクトリ/service の作成

```
# mkdir /service
# chmod 755 /service
```

◇ svscan の起動

ここでサーバーを reboot するか、次のコマンドを実行すると、デーモン管理プログラム svscan が起動する。

```
# csh -cf '/usr/daemontools/bin/svscanboot &'
```

3.3. tcpserver のインストール

◇ make と setup

```
% cd ~/wtmp
% tar xfvz ../src/ucspi-tcp-0.88.tar.gz
% patch -p1 < ../patches/ucspi-tcp-0.88.errno.patch
% make
% su
# make setup check
```

これにより tcpserver を始めとして tcprules、tcprulescheck などのプログラムが/usr/local/bin にインストールされる。

インストール先を変更したいときは、make の前にファイル conf-home を編集し /usr/local を/other_dir などと変更する。そうするとインストール先が/other_dir/bin となる。この変更方法は、daemontools を除く djb ツールに対して通用する。

4. daemontools と tcpserver の使用方法

この節では、djb プログラムの要である daemontools と tcpserver について、それらの概略と使用方法を説明する。具体例として、デーモン sshd を管理する場合を取り上げる。daemontools および tcpserver をそれぞれ単体で使う方法とこれらを連携させて使う方法などいくつか紹介する。

次の URL は、さまざまなサービスに対する設定例が紹介されていて、参考になる。

http://quox.org/tips/server/daemontools.html

4.1. sshd の標準設定の無効化

sshd は Plamo Linux の標準インストールでは、プログラム本体が/usr/sbin にインストールされ、サーバーの起動時に/etc/rc.d/rc.inet2 から呼び出され、デーモンとして起動する設定になっている。そこで、まず標準の設定を無効にする。

◇ sshd の起動スクリプトの無効化

```
# vi /etc/rc.d/rc.inet2
```

起動スクリプト rc.inet2 を編集し、sshd の起動に関係する行の先頭に#を付ける。

rc.inet2 の該当部分

```
#if [ -x /usr/local/sbin/sshd ]; then
#  echo -n " sshd"
#  /usr/local/sbin/sshd
#elif [ -x /usr/sbin/sshd ]; then
#  echo -n " sshd"
#  /usr/sbin/sshd
#fi
```

◇ 動作中のデーモン sshd の停止

```
# ps ax | grep sshd
# kill sshd_pid
```

4.2. daemontools のサービス管理方式

daemontools を使用していて最も便利に感じることは、デーモンを kill したり、デーモンに HUP シグナルを送ったりするときに、そのプロセス ID を調べる必要がないということである。とくに、一つのプログラムで複数の関連するデーモンが同時に稼動しているとき、そう感じる。

daemontools はサービス（デーモン）を管理するためのいくつかのプログラムからなる。

daemontools によるサービス管理は、図 4.1 のように二層構造になっている。svscan はサービスごとに supervise デーモンの起動と管理を行い、supervise は割当てられたサービスの管理を行なう。

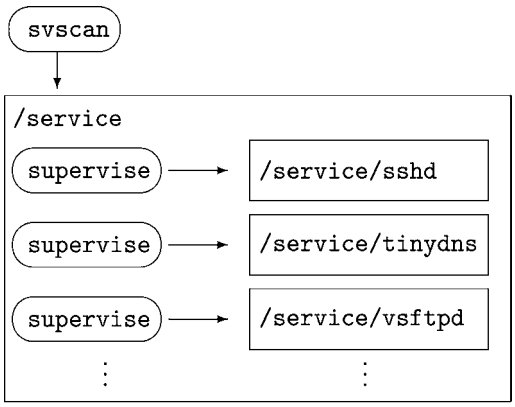

図 4.1 svscan と supervise

サーバー管理者がサービス管理のために使うコマンドとして、svc や svstat などがある。svc はサービスの起動と停止などに用い、svstat はサービスの状態を表示するために用いる。

Service という名前のサービスを daemontools によって管理させるときの手順および経過は次の通りである。

(1) *Service* の管理用ディレクトリ /var/*Service* を作成する。サービスの名前とディレクトリ名は必ずしも一致しなくてもよい。

(2) /var/*Service* に *Service* を起動するためのスクリプト run を作成する。

(3) /var/*Service* から/service にリンクを張る。

(4) 常に/service を監視している svscan は、/service/*Service* の存在を察知して、直ちに *Service* を管理するためのデーモン supervise を起動する。

(5) supervise は管理用ディレクトリ /service/*Service*/supervise を作成する。

(6) supervise は/service/*Service*/run を実行することにより、*Service* を起動する。何らかの理由で *Service* が停止してしまったら、supervise は再度 run スクリプトを実行して *Service* を起動する。

(7) さらに/service/*Service*/log/run が存在するときは、*Service* のログ収集用の supervise デーモンが立ち上がり、log ディレクトリ下でログの収集が行なわれる。

4.3. daemontools による sshd の起動

◇ sshd の管理用ディレクトリの作成

```
# mkdir /var/sshd
```

◇ sshd の起動スクリプトの作成

```
# vi /var/sshd/run
# chmod +x /var/sshd/run
```

/run/sshd/run

```
#!/bin/sh
exec 2>&1
exec env - PATH="/usr/sbin:$PATH" \
sshd
```

◇ /service へリンクをはる

```
# ln -s /var/sshd /service
```

◇ 動作確認

```
# PATH="/usr/daemontools:$PATH"
# export PATH
# svstat /service/sshd
/service/sshd:  up (pid 123) 21 seconds
```

外部からサーバーに slogin でログインできるかも確認する。

4.4. daemontools のコマンド

daemontools の代表的なコマンドである svc や svstat の実行例をまとめておく。

daemontools をインストールしたディレクトリ/usr/daemontools にパスが通っているものとする。

◇ 特定のサービスの状態を調べる

```
# svstat /service/sshd
/service/sshd:  up (pid 123) 21 seconds
```

sshd を管理している supervise のプロセス ID とサービスが始まってからの秒数が表示される。

◇ すべてのサービスの状態を調べる

```
# svstat /service/*
/service/clockspeed:  up (pid 121) 85 seconds
/service/dnscache:  up (pid 111) 85 seconds
/service/httpd:  up (pid 116) 85 seconds
/service/pop3:  up (pid 122) 85 seconds
/service/qmail-send:  up (pid 125) 85 seconds
/service/sshd:  up (pid 123) 85 seconds
/service/tinydns:  up (pid 117) 85 seconds
```

◇ デーモンに HUP 信号を送る (デーモンの config ファイルなどの更新時)

```
# svc -h /service/sshd
```

/etc/sshd.conf を更新したとき、このコマンドにより更新内容が反映される。

◇ デーモンを停止する

```
# svc -d /service/sshd
```

サーバーの reboot 後も停止しておきたいときは、さらにファイル down を作成しておく。

```
# touch /service/sshd/down
```

◇ デーモンを再起動する

```
# rm /service/sshd/down
# svc -u /service/sshd
```

◇ デーモンに TERM 信号を送る (run スクリプトの更新時)

```
# svc -t /service/sshd
```

-t オプションは-d と似ているが、-d では指示を出さない限り、サービスがそのまま停止しているのに対し, -t では/service/sshd が存在すれば、5 秒以内に run スクリプトが実行され、サービスが起動する。したがって、run スクリプトを更新したいときに使用するとよい。

run スクリプトを更新するときは、run.new などのファイル名で作成しておき、

```
# mv run.new run
```

を実行するのが安全である。

◇ サービスを完全に削除する

```
# cd /service/sshd
# rm -rf /service/sshd
# svc -dx .  log
```

4.5. tcpserver による sshd の起動 (アクセス制御なし)

tcpserver には 2 つの機能がある。一つは、外部からの接続要求に対してネットワーク・サービス (デーモン) を起動させること、もう一つは、アクセス制御をすることである。したがって、従来から使われている inetd と TCP Wrapper を組合わせた機能に匹敵する。

inetd との違いは、inetd が複数のサービスの管理をするのに対し、tcpserver は一つのサービスのみを管理することである。それゆえ、複数のサービスの管理には、tcpserver をサービスの数だけ起動する必要がある。この管理方式は、「一つのことをうまくやる」という Unix の哲学に則っているとも言える。

◇ tcpserver コマンドの書式

```
tcpserver [<オプション>] <ホスト>
<ポート番号> <プログラム>
```

<ホスト>には、eth0, eth1 などのネットワーク・インターフェイスに割り当てられた IP アドレスを指定する。インターフェイスが 1 つだけのときや、すべてのインターフェイスを指定したいときは、<ホスト>に 0 を書く。

<ポート番号>には、サービスに割り当てられたポート番号、または、/etc/services に

あるサービス名を指定する。

<プログラム>には、tcpserver 経由で起動したいサービスを指定する。サービス起動時のオプションがあれば、それも一緒に指定する。

◇ tcpserver で ssh を起動する

```
# tcpserver -v 0 22 sshd -i &
```

-v は詳細なメッセージを表示するための tcpserver のオプション。また、-i は sshd のオプションであり、inetd(=tcpserver) からの起動を意味する。

4.6. tcpserver による sshd の起動（アクセス制限あり）

tcpserver によりアクセス制限をするには、以下の例に示すようなルールファイル sshd.rule を作成し、これを cdb 形式ファイル sshd.cdb に変換して、tcpserver の起動時に読み込ませればよい。

sshd.rule の例

```
192.168.1.:allow
1.2.3.80:deny
1.2.3.:allow
:deny
```

ルールファイルでは、最初に該当した行のルールが有効になる。上の例では次のようになる。

- 192.168.1.* の形式の IP アドレスからの接続は許可する。
- 1.2.3.80 からの接続は拒否する。
- 1.2.3.80 以外の 1.2.3.* の形式の IP アドレスからの接続は許可する。
- それ以外の IP アドレスからの接続は拒否する。

◇ ルールファイルの作成

```
# mkdir /var/sshd
# cd /var/sshd
# vi sshd.rule
```

◇ ルールファイルの cdb 形式への変換

```
# tcprules sshd.cdb tmpf < sshd.rule
```

tmpf は一時的に使用するファイルなのでファイル名は何でもよい。

◇ tcpserver による起動

```
# tcpserver -v -x /var/sshd/sshd.cdb
   0 22 sshd -i &
```

-x オプションによりアクセス制御のルールファイル sshd.cdb を読み込んでいる。この部分だけ、アクセス制限なしの場合と異なる。

4.7. ルールファイルのテスト

ルールファイルの設定を確認するには、tcprulescheck コマンドを cdb 形式のファイルに適用する。接続元として環境変数 TCPREMOTEIP に任意の IP アドレスを与えると、どのルールが適用されて、どのようなアクション (許可、不許可) になるのかが表示される。以下は上で挙げた sshd.cdb に対する実行例である。

```
% cd /service/sshd
% TCPREMOTEIP=1.2.3.80 tcprulesche
 ck sshd.cdb
rule 1.2.3.80:
deny connection
% TCPREMOTEIP=1.2.3.40 tcprulesche
 ck sshd.cdb
rule 1.2.3.:
allow connection
```

◇ ルールファイルのコメント行

ルールファイルでは、先頭が#で始まる行はコメント行と見なされ無視される。

4.8. daemontools と tcpserver の連携による sshd の起動 (アクセス制限あり)

daemontools と tcpserver を連携して sshd を管理するには、4.3 と 4.6 で行なった操作を組合わせればよい。すなわち、4.3 で例示した

run スクリプトにおいて、sshd の起動部分を 4.6 で示した tcpserver による起動コマンドに置き換えればよい。

/var/sshd/run

```
#!/bin/sh
exec 2>&1
exec env - PATH="/usr/sbin:$PATH" \
tcpserver -v -x /var/sshd/sshd.cdb \
0 22 sshd -i
```

◇ 実行手順

```
# mkdir /var/sshd
# cd /var/sshd
# vi run
# chmod +x run
# vi sshd.rule
# tcprules sshd.cdb tmpf < sshd.rule
# ln -s /var/sshd /service
```

4.9. ログの収集

daemontools にはログ収集を行なうプログラム multilog が用意されている。サービス *Service* のログ収集を行なうには、次のことを行う。

1. ログ用のグループとユーザーを登録する。
2. ログ用のディレクトリ/var/*Service*/log を作成し、その所有者をログ用ユーザーとする。
3. multilog をログ用のユーザーの権限で起動させるスクリプト /var/*Service*/log/run を作成する。

◇ ログ収集の例

以下の例は、ログ用のグループを slog、ユーザーを dlog とした場合に、sshd のログ収集を行なう手順である。ログ収集を有効にするには、daemontools による sshd 管理をいったん完全に削除してから起動し直す。

```
# groupadd slog
# useradd -g slog dlog
# mkdir /var/sshd/log
# cd /var/sshd/log
# chown dlog /var/sshd/log
# vi run
# chmod +x run
# cd /service/sshd
# rm -rf /service/sshd
# svc -dx .
# ln -s /var/sshd /service
```

/var/sshd/log/run

```
#!/bin/sh
exec setuidgid dlog multilog ./main
```

multilog が起動すると、ディレクトリ /service/*Service*/log/main が作成され、この下のファイル current にログが記録される。時刻を可読形式にするには、daemontools の tai64nlocal コマンドを用いる。

```
# cd /service/sshd/log/main
# cat current
```

```
@40000000435eda7701fd8b5c tcpserver: status: 1/40
@40000000435eda77020167a4 tcpserver: pid 15035 from
 192.168.1.14
@40000000435eda7707a13964 tcpserver: ok 15035 mc.ge
n.nusd.jp:1.2.3.55:22 :192.168.1.14::1036
```

```
# cat current | tai64nlocal
```

```
2005-10-26 10:22:53.033393500 tcpserver: status: 1
/40
2005-10-26 10:22:53.033646500 tcpserver: pid 15035
 from 192.168.1.14
2005-10-26 10:22:53.128006500 tcpserver: ok 15035
mc.gen.nusd.jp:1.2.3.55:22 :192.168.1.14::1036
```

◇ root にならずにログを読む方法

ログ用のディレクトリ main のグループ属性を変更し、グループ slog にユーザー hiro を所属させる。そうすると、ユーザー hiro のままでログを読むことができる。

```
# chmod g+rx /var/sshd/log/main
# vi /etc/group
```

(slog の行に hiro を書き加える)

5. DNS — djbdns

djbdns は、ドメイン名の解決と自ドメインの情報管理を行なう DNS サーバーである。一般的に使用されている BIND に比べ、各機能を別々のプログラムで動作させているのが特徴であり、設定ファイルの書式もわかりやすい。

djbdns は、次の 3 つのサービスに分かれる。

- dnscache — 自ホスト用にドメイン名の解決をするキャッシュ・サービス
- dnscachex — クライアントからの問合せに応える外部キャッシュ・サービス
- tinydns — 権限を委譲された自ドメインの情報を管理する DNS サービス

ここでは、dnscache と tinydns をインストールする。daemontools が既にインストールされていることを前提とする。

なお、dnscachex と tinydns は別々のネットワーク・インターフェイスで動かさなければならない。

5.1. djbdns のインストール

◇ make と setup

```
% cd ~/wtmp
% tar xfvz ../src/djbdns-1.05.tar.gz
% cd djbdns-1.05
% patch -p1 < ../patches/djbdns-1.05
 .errno.patch
% make
# make setup check
```

この操作で、/usr/local/bin に dnscache、tinydns、dnsq などがインストールされる。また、ルートサーバーの IP アドレスが /etc/dnsroots.global に書き出される。

5.2. dnscache の設定

◇ グループとユーザーの登録

dnscache 用のグループ dns とそこに属するユーザー dnscache、dnslog を登録する。

```
# groupadd dns
# useradd -g dns dnscache
# useradd -g dns dnslog
```

ユーザー ID を特定の番号 (例えば 801) に指定したいときは、次のように -u オプションを付ける。

```
# useradd -u 801 -g dns dnscache
```

◇ dnscache 用のデーモン管理

```
# dnscache-conf dnscache dnslog
 /var/dnscache
# ln -s /var/dnscache /service
```

◇ ネームサーバーに localhost を追加

```
# vi /etc/resolv.conf
```

resolv.conf を編集して、最初に問合せるネームサーバーが localhost(=127.0.0.1) になるように指定する。

resolv.conf

```
search gen.nusd.jp nusd.jp
nameserver 127.0.0.1
nameserver 1.2.3.4
```

◇ dnscache の動作確認

dnscache のログはファイル

/var/dnscache/log/main/current

に記録されるので、DNS 検索を行なってみて、このファイルに書き込みが行なわれるかを調べればよい。記録された時刻を見やすく表示するには、daemontools の tai64nlocal コマンドを用いる。

```
# PATH="/usr/daemontools:$PATH"
# export PATH
# cd /var/dnscache/log/main
# tail -f current | tai64nlocal
```

別の端末を開き次のコマンドを実行する。最初の端末で何かが表示されれば、正常動作し

ている。

```
# dnsip ns.nusd.jp
1.2.3.4
```

5.3. tinydns の設定

tinydns は、権限の委譲されたドメイン情報の管理を行なう。ここではその設定をする。ドメイン情報は第 2 節で列挙したもの (表 2.2, 2.3, 2.4 参照) を仮定する。

◇ ユーザーの登録

```
# useradd -g dns tinydns
```

◇ tinydns の管理用ディレクトリの作成

```
# tinydns-conf tinydns dnslog
 /var/tinydns 1.2.3.55
```

◇ ゾーン情報のテキスト・ファイル作成

add-xxx という形のコマンドを実行し、ドメイン情報をファイル data に書き出していく。

```
# cd /var/tinydns/root
# ./add-ns gen.nusd.jp 1.2.3.55
```

ドメイン gen.nusd.jp を管理するサーバーの IP アドレスは 1.2.3.55。

```
# ./add-host mc.gen.nusd.jp 1.2.3.55
```

ホスト名 mc.gen.nusd.jp の IP アドレスは 1.2.3.55。

```
# ./add-alias www.gen.nusd.jp 1.2.3
 .55
```

IP アドレス 1.2.3.55 をもつホストの別名は www.gen.nusd.jp。

```
# ./add-mx gen.nusd.jp 1.2.3.7
```

ドメイン gen.nusd.jp 宛のメールは IP アドレス 1.2.3.7 のホストで受ける。

```
# ./add-ns gen.3.2.1.in-addr.arpa
 1.2.3.55
```

逆引き用のドメイン gen.3.2.1.in-addr.arpa を管理するサーバーの IP アドレスは 1.2.3.55。

```
# vi data
```

逆引き用のホスト名を登録するコマンドはないので、エディタで data を編集する (ファイルの最終行を追加)。

data

```
.gen.nusd.jp:1.2.3.55:a:259200
=mc.gen.nusd.jp:1.2.3.55:86400
+www.gen.nusd.jp:1.2.3.55:86400
@gen.nusd.jp:1.2.3.7:a::86400
.gen.3.2.1.in-addr.arpa:1.2.3.55:a:259200
^55.gen.3.2.1.in-addr.arpa:mc.gen.nusd.jp
```

◇ data のコメント行

data ファイルでは、先頭が#で始まる行はコメント行と見なされ無視される。

◇ ドメイン情報を cdb 形式に変換

```
# make
```

これでテキスト・ファイル data を cdb 形式に変換したファイル data.cdb が作成される。

◇ tinydns の起動

```
# ln -s /var/tinydns /service
```

◇ 動作確認

```
# dnsq any gen.nusd.jp 1.2.3.55
```

```
255 gen.nusd.jp:
163 bytes, 1+3+0+2 records, response, authoritati
ve, noerror
query: 255 gen.nusd.jp
answer: gen.nusd.jp 2560 SOA a.ns.gen.nusd.jp hos
tmaster.gen.nusd.jp 1129536494 16384 2048 1048576
2560
answer: gen.nusd.jp 259200 NS a.ns.gen.nusd.jp
answer: gen.nusd.jp 86400 MX 0 a.mx.gen.nusd.jp
additional: a.ns.gen.nusd.jp 259200 A 1.2.3.55
additional: a.mx.gen.nusd.jp 86400 A 1.2.3.7
```

```
# dnsname 1.2.3.55
mc.gen.nusd.jp
```

◇ data ファイルの修正

上記の設定では、

NS レコード　a.ns.gen.nusd.jp

MX レコード　a.mx.gen.nusd.jp

というように、自動的に意図しない情報も設定されてしまう。これでも動作に支障はないようだが、これらの名前を使いたくない場合

は、data を次のように編集する。以下の例ではメールサーバーの preference（優先順位）を 10 に指定している。

```
# vi data
```

data の編集例

```
.gen.nusd.jp::mc.gen.nusd.jp:259200
=mc.gen.nusd.jp:1.2.3.55:86400
+www.gen.nusd.jp:1.2.3.55:86400
.gen.3.2.1.in-addr.arpa::mc.gen.nusd.jp:
259200 （前行に続く）
@gen.nusd.jp:1.2.3.7:mail.nusd.jp:10:86400
```

◇ data ファイルの修正を反映

```
# make
```

◇ 修正後の動作確認

```
# dnsq any gen.3.2.1.in-addr.arpa
 1.2.3.55

255 gen.3.2.1.in-addr.arpa:
152 bytes, 1+2+0+1 records, response, authoritati
ve, noerror
query: 255 gen.3.2.1.in-addr.arpa
answer: gen.3.2.1.in-addr.arpa 2560 SOA mc.gen.nu
sd.jp hostmaster.gen.3.2.1.in-addr.arpa 112953987
1 16384 2048 1048576 2560
answer: gen.3.2.1.in-addr.arpa 86400 NS mc.gen.nu
sd.jp
additional: mc.gen.nusd.jp 86400 A 1.2.3.55
```

5.4. djbdns の検索ツール

◇ IP アドレスを調べる

```
% dnsip mc.gen.nusd.jp
1.2.3.55
```

◇ ホスト名を調べる

```
% dnsip 1.2.3.7
mail.nusd.jp
```

◇ メールサーバーを調べる

```
% dnsmx gen.nusd.jp
10 mail.nusd.jp
```

◇ サーバを指定して各種情報を調べる

書式　dnsq <type> <name> <server>

```
% dnsq ns gen.nusd.jp 1.2.3.55
```

この例では、DNS サーバーに 1.2.3.55 を指定してドメイン gen.nusd.jp のネームサーバーを調べている。

<type>には any, ns, mx, a などが指定可能。

6. SMTP — qmail

qmail は旧世代の sendmail に置き換わるメールサーバーである。sendmail がすべての機能をひとつの巨大なプログラムで実行するのに対し、qmail はいくつかの小さなプログラムからなり、それぞれが固有の役割を担う。また、root 権限で動作するプログラムを極力少なくすることにより、セキュリティを高めている。

大まかに分類すれば、qmail-send とその仲間がローカル→ローカル、ローカル→外部の配送を行い、qmail-smtpd が外部→ローカルの配送を行なう。したがって、以下のインストール作業でも、起動設定の作業は qmail-send 用と qmail-smtpd 用の 2 つが必要になる。qmail-send がデーモンとして起動されると、qmail-lspawn、qmail-rspawn、qmail-clean も一緒にデーモンとして立ち上がる。

qmail-smtpd をアクセス制限なしで起動すると、メールの不正リレーに悪用されるので注意が必要である。

qmail について参考となる Web ページは、本家 Bernstein 氏のページ

```
http://cr.yp.to/qmail.html
```

の他に次のものがある。

```
http://www.lifewithqmail.org/lwq.html
http://www9.jp.qmail.org/top.html
```

6.1. qmail のインストール

◇ ソースの展開とパッチあて

```
% cd ~/wtmp
% tar xfvz qmail-1.03.tar.gz
% cd qmail-1.03
```

```
% patch -p1 < ../patches/qmail-1.03
.errno.patch
% patch -p1 < ../patches/qmail-1.03
.qmail_local.patch
```

◇ ディレクトリの作成

```
# mkdir /var/qmail
```

◇ グループとユーザーの登録

```
# groupadd nofiles
# useradd -g nofiles -d /var/qmail/alias
alias
# useradd -g nofiles -d /var/qmail qmaild
# useradd -g nofiles -d /var/qmail qmaill
# useradd -g nofiles -d /var/qmail qmailp
# groupadd qmail
# useradd -g qmail -d /var/qmail qmailq
# useradd -g qmail -d /var/qmail qmailr
# useradd -g qmail -d /var/qmail qmails
```

◇ プログラムのインストール

```
# make setup check
```

これにより、qmail のプログラム群が /var/qmail 以下にインストールされる。

また、メールキューなどのディレクトリも /var/qmail の下に作られる。

◇ control ファイルの作成

```
# ./config
```

DNS について正引き、逆引きが正しく設定されていれば、/var/qmail/control に me、locals、rcpthosts、defaultdomain、plusdomain が自動生成される。自動作成に失敗しても、エディタで作成すればよい。

サーバーの使用環境に合わせて、これらのファイルを編集し、さらに、必要ならば defaulthost, smtproutes も作成するとよい。とくに、smtproutes は、外部に送信するメールをウィルスチェック・サーバーに通すために必要となる。

◇ control ファイルの内容と説明

• **control/me**

内容 mc.gen.nusd.jp

説明 サーバーのホスト名（ドメイン名も含む）

• **control/locals**

内容 localhost
mc.gen.nusd.jp
gen.nusd.jp

説明 qmail-send が参照。このファイルをもとに、メールがローカル宛と外部宛とに振り分けられる。このファイルを更新したら qmail-send に HUP シグナルを送る。

• **control/rcpthosts**

内容 localhost
mc.gen.nusd.jp
gen.nusd.jp

説明 qmail-smtpd が参照。外部から来たメールのうち、宛先アドレスの@以降の部分がこのファイルに記述された行のどれかに一致していたら、そのメールを受け取る。

• **control/smtproutes**

内容 mc.gen.nusd.jp:[1.2.3.55]
gen.nusd.jp:[1.2.3.55]
:[1.2.3.7]

説明 送信メールの宛先の@以降の部分が左辺に一致したら、右辺の IP アドレスのホストに送信する。この例では、外部宛のメールはすべて学部のメールサーバー (1.2.3.7) に送る設定になっている。

• **control/defaulthost**

内容 gen.nusd.jp

説明 qmail-inject が参照。宛先および送信者のメールアドレスの@以降が省略されている場合に付加する文字列。例えば、hiro は hiro@gen.nusd.jp に変換される。

• **control/defauldomain**

内容 gen.nusd.jp

説明 qmail-inject が参照。宛先および送信者のメールアドレスの@以降でドット (.) が省略されている場合に付加する文字列。例えば、

hiro@mc は hiro@mc.gen.nusd.jp に変換される。

• **control/plusdomain**

内容 nusd.jp

説明 qmail-inject が参照。宛先のメールアドレスが+で終わっているときに、付加する文字列。例えば、hiro@mc.gen+は hiro@mc.gen.nusd.jp に変換される。

◇ alias file の作成

```
# cd ~alias
# touch .qmail-postmaster
# touch .qmail-mailer-daemon
# touch .qmail-root
# touch .qmail-hostmaster
# touch .qmail-webmaster
# chmod 644 .qmail*
```

tinydns で自動的に設定される hostmaster や apache などの WWW サーバーで設定する webmaster の alias も作っておく。

postmaster 宛のメールをユーザー hiro で受け取りたいときは、次を実行する。

```
# echo hiro > .qmail-postmaster
```

◇ 一般ユーザー用のメールボックスの作成

maildir 形式のメールボックスを一般ユーザー用に作成する。以下の例はユーザー hiro の場合。

```
# su hiro
% whoami
hiro
% /var/qmail/bin/maildirmake ~/mail
dir
% echo ./maildir/ > ~/.qmail
% exit
```

この操作で/home/hiro に maildir と maildir/new、maildir/cur、maildir/old が作成される。また、~hiro/.qmail に「./maildir/」と記述されているので、hiro 宛のメールは~hiro/maildir/ 以下のディレクトリに格納される。

◇ qmail-send 起動ファイルの作成

```
# cd /var/qmail
# cp boot/home ./rc
# vi rc
```

maildir 形式のメールボックスを使うためにファイル rc を編集し、文字列./Mailbox を文字列./Maildir/ に置き換える。

◇ 古い smtp の無効化

Plamo Linux では SMTP サーバーとして postfix がインストールされるので、postfix を丸ごと削除する。

```
# removepkg postfix
```

postfix を残しておきたいときは、removepkg コマンドの代わりに次を実行する。

```
# chmod 0 /usr/sbin/sendmail
# mv /usr/sbin/sendmail \
> /usr/sbin/sendmail.bak
```

さらに、inetd の設定の comstat の行を無効にする。

```
# vi /etc/inetd.conf
# ps aux | grep inetd
# kill -HUP inetd_pid
```

◇ sendmail のリンク作成

```
# ln -s /var/qmail/bin/datemail
/usr/sbin/sendmail
```

/var/qmail/bin には sendmail と datemail という同等な機能を持つプログラムがあるが、datemail のほうは、ローカルタイムで時刻を記録するので、こちらをリンク先に指定する。

6.2. サーバー起動時の標準的な設定

◇ qmail-send の設定

/etc/rc.d/rc.local に次の行を追加する。

```
csh -cf '/var/qmail/rc &'
```

◇ qmail-smtpd の設定 (アクセス制限なし)

ユーザー qmaild のグループ ID(nofiles) とユーザー ID を調べる。

```
# grep qmaild /etc/passwd
```

コロン区切りの項目のうち、3 番目がユーザー ID で、4 番目がグループ ID。

ユーザー qmaild のユーザー ID とグループ ID がそれぞれ 812、810 であるとすれば、/etc/rc.d/rc.local に次の行を追加すればよい。

```
/usr/local/bin/tcpserver -u 812 -g
810 0 smtp /var/qmail/bin/qmail-smtpd
&  (1 行で書く)
```

◇ qmail-smtpd の設定 (アクセス制限あり)

まず、アクセス制限のためのルールファイル smtpd.rule (下の例参照) を作成して、それを cdb 形式に変換する。

```
# cd /var/qmail/control
# vi smtpd.rule
# tcprules smtpd.cdb tmpf < smtpd.rule
```

そして、/etc/rc.d/rc.local の smtpd の行を次で置き換える。

```
/usr/local/bin/tcpserver -x /var/qma
il/control/smtpd.cdb -u 812 -g 810 0
smtp /var/qmail/bin/qmail-smtpd &
                              (1 行で書く)
```

smtpd.rule の例

```
127.0.0.1:allow,RELAYCLIENT=""
192.168.1.:allow,RELAYCLIENT=""
:allow
```

この例では次のようなルールになっている。

- SMTP クライアントからの配送依頼は自ホストと eth1 側のネットワークからのみ許可する。
- すべての SMTP サーバーからのメール配送は許可する。

6.3. daemontools による qmail の管理

qmail-send と qmail-smtpd の 2 つに対して行なう。supervise による管理ディレクトリはそれぞれ

```
/var/qmail/qmail-send
/var/qmail/qmail-smtpd
```

とする。

設定する前に、動作中の

qmail-send, splogger qmail, qmail-lspawn,
qmail-rspawn, qmail-clean,
qmail-stmpd(tcpserver から起動)

はすべて停止させる。さらに、/etc/rc.d/rc.local の qmail-send, qmail-smtpd の設定行は無効にしておく。

◇ qmail-send 用の設定

```
# mkdir /var/qmail/qmail-send
# cd /var/qmail/qmail-send
# vi run
# chmod +x run
# ln -s /var/qmail/qmail-send
 /service
```

/var/qmail/qmail-send/run

```
#!/bin/sh
echo 'Starting qmail...'
exec env - PATH="/var/qmail/bin:$PATH" \
qmail-start ./Maildir/ splogger qmail
```

splogger によりログは syslog に渡され、/var/log/message に記録される。

◇ qmail-smtp 用の設定

6.2 で作成したアクセス制限のためのルールファイルは、/var/qmail/qmail-smtpd に移して使用することにする。

```
# mkdir /var/qmail/qmail-smtpd
# cd /var/qmail/qmail-smtpd
# vi run
# chmod +x run
# mv ../control/smtpd.rule .
```

```
# mv ../control/smtpd.cdb .
# ln -s /var/qmail/qmail-smtpd
 /service
```

/var/qmail/qmail-smptd/run

```
#!/bin/sh
echo 'Starting qmail-smtpd...'
exec env - PATH="/var/qmail/bin:$PATH" \
envuidgid qmaild \
tcpserver -x /var/qmail/qmail-smtpd/smtpd.cdb \
0 smtp qmail-smtpd 2>&1
```

envuidgid によりユーザー ID とグループ ID を qmaild のものに設定して tcpserver を起動している。

6.4. ウィルスチェックのための設定

既に述べたように、メールのウィルスチェックをするために、外部とのメール送受信は学部のメール・サーバー (表 2.3 参照) を介して行なっている。そのために必要な設定を整理しておく。

◇ 外部からの受信

DNS の設定で gen.nusd.jp ドメインの MX レコードを学部のメールサーバーに指定する。

/var/tinydns/root/data の該当箇所

```
@gen.nusd.jp:1.2.3.7:mail.nusd.jp:10:86400
```

◇ 外部への送信

/var/qmail/control/smtproutes を作成し、自ホストと自ドメイン gen.nusd.jp 宛のメールは自ホスト (1.2.3.55) で受け、それ以外のメールはすべてウィルスチェック・サーバー (1.2.3.7) に送るように設定する。

/var/qmail/control/smtproutes

```
mc.gen.nusd.jp:[1.2.3.55]
gen.nusd.jp:[1.2.3.55]
:[1.2.3.7]
```

7. 時刻合わせ

サーバーの設定で見落としやすいのが時刻調整である。正しい時刻に設定されていないと、サーバーの利用者や他のサーバーに迷惑をかけるので注意したい。

サーバーの内蔵時計はかなり不正確なようだ。手元のマシンでの計測によれば、一日に 9 秒ほどずれてしまう。一ヶ月にすると 4 分から 5 分の狂いが生じる計算だ。

一般には、xntp が時刻合わせのために利用されているが、ここでは djb ツールの一つである clockspeed を用いる。xntp が定期的に上位のタイムサーバーと同期をとるのに対し、clockspeed は、タイムサーバーとの同期を数回程度行なえば、後はタイムサーバーに頼ることなく自立して正確な時刻を刻むことができる。その理由は次項で説明する。

なお、clockspeed のソースに同梱されている taiclock を使えば、サーバーにタイムサーバーの機能を持たせることができる。

7.1. clockspeed の動作原理

マシンには一定のリズムで動作するものが 2 つ存在する。一つは内蔵時計であり、もう一つは CPU である。もちろん、内蔵時計の示す時刻がそのマシンで利用される時刻となる。たとえば、date コマンドで表示される時刻や、各種サービスのログに記録される時刻がそうである。一方、CPU の速さは 1 秒間に実行できる命令（または処理）の回数（クロック数）で決まる。100MHz の CPU ならば、1 秒間に $100 \times 10^6 = 1$ 億 回の命令を行なう。したがって、時刻を測るもう一つの量として、電源を入れてからの CPU の総クロック数を用いることができる。

さて、時刻（正確な時刻のこと）が t のときの、内蔵時計の時刻を $I(t)$、電源投入後の CPU の総クロック数を $C(t)$ で表すことにす

る。内蔵時計も CPU も一定のリズムで動作するから、式

$$\frac{I(t_2)-I(t_1)}{t_2-t_1}, \qquad \frac{C(t_2)-C(t_1)}{t_2-t_1}$$

で求められる量はどちらも、2つの時刻 t_1, t_2 によらずに一定であり、それぞれ単位時間あたりの $I(t)$ および $C(t)$ の増加量を表している。

時刻 t における内蔵時計の時間のずれは、$I(t)-t$ と表されるから、CPU の1クロックあたりの時間のずれ d は

$$d=\frac{(I(t_2)-t_2)-(I(t_1)-t_1)}{C(t_2)-C(t_1)}$$

と計算することができる。したがって、定数 C_0 を一つ定めて、CPU の総クロック数が C_0 増すごとに内蔵時計の時刻を

$$C_0 \times d$$

だけ補正して ($I(t_0)$ から d を減じて) いけば、内蔵時計の時間のずれを一定のまま保つことができる。つまり、正確な時刻を保持しているタイムサーバーへ時刻の問合せを少なくとも2回（時刻 t_1 と時刻 t_2 に）行なうだけで、内蔵時計の正確な時刻合わせが可能となる。

clockspeed の実際の運用では次のようにする。まず、時刻 t_0 で $I(t_0)=t_0$ となるように内蔵時計の時刻合わせをする。そして、あまり間をおかずに時刻 t_1 で最初のずれを計測する。2時間後（時刻 t_2）に再びずれの計測を行なう。以降、2日後、2週間後、2ヵ月後というように指数関数的な間隔でずれの計測を行なう。計測の間隔（$=t_2-t_1$）を大きくしていくのは、t_2-t_1 が大きいほど、内蔵時計のずれの測定誤差が小さくなるからである。

clockspeed のマニュアルによれば、内蔵時計の動きに狂いがなければ、1世紀にわずか数秒の誤差しかない高精度の時計が得られる。

7.2. at コマンド

前述したように、clockspeed による時刻合わせでは、2時間後、2日後というように時刻合わせのコマンドを実行する必要がある。そのために便利なのが、指定時刻に指定したコマンドを実行してくれる at コマンドである。Linux の標準的なインストールでは、そのためのデーモン atd が起動するようになっている。メールサーバーが動作していないと、at コマンドが有効にならないので注意が必要である。

◇ at の使用例

次はユーザー hiro がホームディレクトリに「hello aboshi!」という内容のテキストファイル `tmp.txt` を5分後に作成する例である。

```
% at +5 minutes
>echo hello aboshi!  > tmp.txt
>  [ctrl+d]
```

at コマンドを入力し、Enter キーを押すと、次の行の先頭に > が現れてライン入力モードになる。ここで指定時刻に実行させたいコマンドを入力し、Enter キーを押す。ライン入力モードから抜け出すには、`ctrl+d` とキー入力すればよい。

at で指定した実行コマンドが標準出力にメッセージを出すタイプのものならば、標準出力の内容がメールで送られてくる。

◇ at の `-f` オプション

`-f` オプションを使えば, 実行コマンドをコマンドラインから入力する代わりに、ファイルから読み込ませることも可能である。

上と同じことを `-f` オプションを使って行なうには、「echo hello aboshi!」という内容のファイル `tmp.sh` を作成して、次を実行すればよい。

```
% at +5 minutes -f tmp.sh
```

at コマンドでは、時間の単位として minutes, hours, days などが使える。

◇ at の関連コマンド

at と関連するコマンドを以下に挙げる。

`atq` 登録されたジョブの一覧表示
`atrm` *job_number* ジョブ番号に対応するジョブを削除する

◇ sleep コマンド

at コマンドが機能しないときは、窮余の策として、sleep コマンドを使うことができる。ただし、すべてのコマンドが終了するまで、その端末は使用できない。次は、10 分後に *command1* を実行し、さらに 2 日後に *command2* を実行する例である。

```
% sleep 10m; command1; sleep 2d;
 command2
```

7.3. clockspeed の設定

◇ make と setup

インストール先を/usr/clockspeed に変更するために、make の前にファイル conf-home の /usr/local/clockspeed を /usr/clockspeed に変更しておく。

```
% cd /wtmp
% tar xfvz clockspeed-0.62.tar.gz
% cd clockspeed-0.62
% patch -p1 < ../patches/clockspeed
 -0.62.errno.patch
% vi conf-home
% make
# make setup check
```

これで clockspeed のプログラム群が /usr/clockspeed にインストールされる。

◇ タイムサーバーを利用した時刻調整

以下では、学部のタイムサーバーが1.2.3.8であると仮定する。

```
# PATH="/usr/clockspeed/bin:$PATH"
# export PATH
# sntpclock 1.2.3.8 | clockview
before: 2005-10-18 18:56:21.335025000000000000
after:  2005-10-18 18:50:55.997481293321378528
```

タイムサーバーを利用して時刻合わせをするとしたら、その前後で内蔵時計の時刻がどうなるかが表示される。実際の調整は行なわれない。上の表示例では、5 分 30 秒ほどずれている。

実際の時刻合わせは次のコマンドで行なう。

```
# sntpclock 1.2.3.8 | clockadd
```

再び clockview を行なうとずれが 0.01 秒未満になる。

```
# sntpclock 1.2.3.8 | clockview
before: 2005-10-18 18:52:00.631551000000000000
after:  2005-10-18 18:52:00.633568011385977268
```

◇ clockspeed の起動

```
# clockspeed &
```

◇ 内蔵時計のずれの測定

次のコマンドを数分後、2 時間後、2 日後、2 週間後、2ヵ月後に実行する。

```
# sntpclock 1.2.3.8 > /usr/clockspe
 ed/adjust &
```

経験によれば、2 時間の間隔で adjust コマンドを 2 回実行するだけで、内蔵時計の時間のずれは 1 日当たり約 0.02 秒であった。1ヶ月当たりに換算すると、約 0.6 秒のずれである。

上に挙げたコマンドは、実際には at コマンドを利用して行なうと便利である。以下に示すファイル adjust.sh を作成しておいて、次のように実行すればよい。

```
# vi adjust.sh
# at +60 days -f adjust.sh
```

```
# at +14 days -f adjust.sh
# at +2 days -f adjust.sh
# at +2 hours -f adjust.sh
# at +3 minutes -f adjust.sh
```

adjust.sh

```
#!/bin
/usr/clockspeed/bin/sntpclock 1.2.3.8 |
/usr/clockspeed/adjust & (前行に続ける)
```

◇ サーバー起動時の設定

サーバー起動時には、タイムサーバーで時刻合わせをした後、clockspeed を起動すればよい。/etc/rc.d/rc.local に次の 2 行を追加する。

```
/usr/clockspeed/bin/sntpclock 1.2.3.8 |
/usr/clockspeed/bin/clockadd (前行に続ける)
/usr/clockspeed/bin/clockspeed &
```

7.4. daemontools による clockspeed の管理

◇ 動作中の clockspeed の停止

```
# ps aux | grep clockspeed
# kill clockspeed_pid
```

◇ /etc/rc.d/rc.local の編集

clockspeed の起動を設定した行を削除する (時刻合わせをする clockadd の行はそのまま)。

```
# vi /etc/rc.d/rc.local
```

◇ clockspeed の管理用ディレクトリの作成

clockspeed の管理用ディレクトリ /var/clockspeed と run スクリプトを作成し、/service にリンクを張る。

```
# mkdir /var/clockspeed
# vi /var/clockspeed/run
# chmod +x /var/clockspeed/run
# ln -s /var/clockspeed /service
```

/var/clockspeed/run

```
#!/bin/sh
exec 2>&1
echo 'Starting clockspeed...'
exec env - PATH="/usr/clockspeed/bin:$PATH" \
/usr/clockspeed/bin/clockspeed
```

謝　辞

本論文の一部は文部科学省学術フロンティア推進事業による.

文　献

1) 日経 Linux 編集部 (2000) 高性能メール配送エージェント qmail 徹底活用. 日経 Linux 3 月号, 82–109

2) 広瀬雄二 (2002) DJB で固めるインターネットサービス. UNIX USER 7 月号, 26–71

3) 広瀬雄二 (2000) FreeBSD で作るセキュアサーバー Part 3 — 安全なメールサーバー／クライアント環境の構築. UNIX USER 12 月号, 65–76

4) 広瀬雄二 (2001) DNS を極める Part 3 djbdns による簡単 DNS 管理. UNIX USER 1 月号, 49–61

5) 広瀬雄二 (2001) Postfix vs. qmail. UNIX USER 4 月号, 34–71