Mac OS X
—コマンドライン・システム管理入門—

内　藤　久　資

7回にわたって書き続けてきたMacOS Xの解説の締めくくりとして，今回はMacOS Xの「コマンドライン」ツールの解説をしたい[1]。

Appleユーザの多くは，そのGUIの操作感のよさや，MacOS Xの安定性・安全性などを理由にMacOS Xを利用しているのだと考える。MacOS Xがシステムとして安定している理由の多くは，MacOS Xの基本システムがBSDと呼ばれるUNIXであることに依存している。MacOS XにはBSDシステムとしてのコマンドラインでのツールが数多く含まれているため，それらを利用することにより，UNIXの手法を使った利用形態が可能である。コマンドラインでの利用法は，特にシステム管理に有効なものが多い。BSDのツール群そのものの解説はFreeBSDなどの解説書などにゆずり，MacOS X特有のツール群のコマンドラインでの利用法に焦点を絞り解説する。

17　コマンドラインの利用法とBSDツール群

はじめに，コマンドラインでMacOS Xを利用するための方法を簡単に述べ，MacOS Xにインストールされている BSDツール群を簡単に紹介しておこう。

17.1　コマンドラインの利用法

「コマンドラインでMacOS Xを利用する」とは，対話型シェルを利用して，コマンドを入力してMacOS Xを利用するということである。そのため，最初に対話型シェルが利用できる「仮想ターミナル」を起動する必要がある。

MacOS Xで標準的に利用できる仮想ターミナルソフトウェアとしては以下のものがある。

- 「ターミナル」アプリケーション
  「アプリケーション」フォルダの中の「ユーティリティ」フォルダにある。「ターミナル」アプリケーションは，日本語表示，UTF-8文字表示にも対応している。
- 「X11」の仮想ターミナルであるxterm，kterm，uxtermなど
  「X11」は，Tiger Install Discを利用するとインストールすることができる。xtermは日本

1　この原稿を執筆している2005年11月現在，MacOS Xの最新リリースは10.4.3である。この解説は，特に断らない限り，MacOS X 10.4.3に沿うものとご理解いただきたい。

語表示には対応していないが，uxterm はUTF-8 に対応している。古典的な日本語仮想ターミナルであるkterm はAppleから配布されているX11 のパッケージには入っていないので，個別にインストールする必要がある。

また，「システム環境設定」の「共有設定」で「リモートログイン」を有効にしている場合には，sshを利用してネットワーク経由でこれらの仮想ターミナルを利用することも可能である。

さて，仮想ターミナルを開くとすぐに対話型シェルが起動する[2]。仮想ターミナルを利用する際にはつぎのことに注意が必要である。

- Aqua 環境では「デスクトップ」などと日本語で表示されていたフォルダ名は，シェル環境では“Desktop”などとなり，Aqua の英語環境で表示されるフォルダ名がディレクトリ名となっている
- ファインダ環境で「フォルダ」として表示されるものは，UNIX システムでのディレクトリであるが，「エリアス」は UNIX システムでのシンボリックリンクではない
- ファイル名に含まれる日本語が表示できない

MacOS X のファイルシステムでは，ファイル名は UTF-8 エンコーディングが利用されている。「ターミナル」アプリケーションを使って `ls` コマンドで日本語を含むファイル名を正しく表示させるには -v オプションをつければよい[3]。

なお，仮想ターミナルを起動した時点では，ログインしているユーザの権限でコマンドが動作することになるが，「管理者権限」が必要な場合には `sudo` コマンドを利用して，管理者モードに移行する必要がある。

【sudo の利用法】

- **`sudo COMMAND`**

  `COMMAND` を管理者権限で実行する。実行後は一般ユーザ権限に戻る

- **`sudo -s`**

  管理者モードに移行する。一般ユーザモードに戻るには `exit` コマンドを実行する

なお，`sudo` を利用して管理者モードに移行できるユーザは「管理者権限」を持つユーザに限られる。この設定を変更するには `sudo` コマンドの動作を規定している `sudoer` ファイルを `visudo` コマンドを利用して書き換えればよい。

17.2 BSD ツール群

冒頭にも述べたとおり，MacOS X は，その裏には BSD システムが存在しているので，基本的な BSD システムのツール群の多くは，デフォルトでインストールされている。どのようなツール群（アプリケーション）がインストールされているかを知るためには，`/usr/bin`，`/usr/`

2　Tiger のデフォルト設定では bash が起動するが，ユーザのログインシェルの設定を変更するとtcsh などの他のシェルを利用することも可能である。

3　コマンドの引数に日本語を含むファイル名を指定することも可能である。

sbin，/usr/X11R6/bin などのディレクトリを覗けばよい。そこには，BSD ユーザならば見たことがあるツール群が一とおりそろっている。

しかしながら，ごく一部に通常の BSD システムと同じ名前であるが，異なった挙動をするコマンドがあるので，それだけはメモしておこう[4]。

- /usr/bin/tar コマンド

  /usr/bin/tar の他に /usr/bin/gnutar がある。/usr/bin/tar を使うと，すでに存在しているディレクトリの上に上書きができない。通常の tar コマンドと同じ動きをするのは /usr/bin/gnutar である。

- /bin/cp，/bin/rm，/bin/mv などのファイル操作コマンド

  これらのコマンドを MacOS X のアプリケーション，またはそれらが作成したファイルに利用するのは危険である[5]。なぜなら，MacOS X のアプリケーションやそれらが作成したファイルには「リソース」と呼ばれる属性データが付随している場合があり，これらのファイル操作コマンド群ではリソースを含んだファイル操作ができない。リソースを含んだファイルに対するファイル操作を行うには「開発環境」に付属するコマンド群を利用する必要がある。

17.3 各種のサービス

MacOS X のネットワークサービスなどに利用されているソフトウェアの多くも，BSD などで標準的に利用されているものが流用されている。それらのコマンドラインなどでの管理方法はそれぞれのウェブページなどを参照すればよい。

★プリンティングシステム

プリンティングシステムは cups が利用されている（cf. [6]）

★ネットワークサービス

ネットワークサービスの制御には xinetd が利用されている（cf. [7]）

★ウェブサーバ

ウェブサーバは apache が，サーブレットコンテナには tomcat が利用されている（cf. [8]）

18 MacOS X 特有のコマンド群

以下では MacOS X に特有なコマンドで，システム管理等に有効なものを順に紹介しよう。それらは GUI ベースのアプリケーションと同等の動きをするものが多く，これらを組合せることにより，システム管理の自動化を実現することができる。

なお，ここで紹介するコマンドの使用例は，各コマンドの機能の一部であり，詳細は man コマンドを利用してオンラインマニュアルを参照していただきたい。

4　これは筆者がハマった例であり，他にも異なった挙動をするコマンドが存在する可能性がある。

5　通常のファイルであれば問題はない。

18.1　ディスクユーティリティ

「ディスクユーティリティ」では以下の操作が可能である。

- ディスクのフォーマット，パーティションの作成，（ソフトウェア）RAID の設定，ディスクの検証などのディスクの管理
- ディスクイメージの作成と復元

これらの操作に対応するコマンドは hdiutil と diskutil である。diskutil はローカルディスクの操作を行うコマンドであり，hdiutil はディスクイメージの操作を行うコマンドである。

18.1.1　ディスクの操作

diskutil を用いるとローカルディスクに対してつぎの操作を行うことができる。

- ディスク情報の取得
- ディスクのマウントとアンマウント
- 取りだし可能なディスクの取りだし
- ディスク名（ボリューム名）の変更
- HFS+ journaring の有効化・無効化
- ディスクのチェックとリペア
- ディスクパーミッションのチェックとリペア
- ディスクやボリュームの消去，再フォーマット
- ディスクパーティションの作成
- RAID の作成・チェック

【使用例】

★ディスクのリストの取得

diskutil list を実行すると以下のような結果を得る。

```
/dev/disk0
   #:                       type name    size       identifier
   0: Apple_partition_scheme          *232.9 GB disk0
   1:    Apple_partition_map          31.5 KB   disk0s1
   2:              Apple_HFS Second  232.8 GB  disk0s3
/dev/disk1
   #:                       type name    size       identifier
   0: Apple_partition_scheme          *149.1 GB disk1
   1:    Apple_partition_map          31.5 KB   disk1s1
   2:         Apple_Driver43          28.0 KB   disk1s2
   3:         Apple_Driver43          28.0 KB   disk1s3
   4:       Apple_Driver_ATA          28.0 KB   disk1s4
   5:       Apple_Driver_ATA          28.0 KB   disk1s5
   6:         Apple_FWDriver          256.0 KB  disk1s6
   7:     Apple_Driver_IOKit          256.0 KB  disk1s7
   8:          Apple_Patches          256.0 KB  disk1s8
   9:              Apple_HFS MacOSX  149.0 GB  disk1s9
```

```
/dev/disk2
   #:                       type name    size       identifier
   0: Apple_partition_scheme          *152.7 GB disk2
   1:          Apple_FWDriver          128.0 KB  disk2s1
   2:    Apple_partition_map           8.0 KB    disk2s2
   3:               Apple_HFS Data     152.7 GB  disk2s3
```

ここに表示されたのは，diskutil が扱うことができるローカルに接続されたディスクのリストと，そのディスクのパーティション情報である。diskutil の各動作で device と指定されているところには，/dev/disk0 などの各ディスクを指定する「デバイス名」を指定する。ここには「ディスクユーティリティ」でディスクとして表示されるものがすべてリストされていることがわかる。

★ディスク情報の取得

diskutil info /dev/disk1 を実行する。すなわち「デバイス」/dev/disk1 の情報を取得すると以下のような結果を得る。

```
Device Node:         /dev/disk1
Device Identifier:    disk1
Mount Point:
Volume Name:

Partition Type:      Apple_partition_scheme
Bootable:            Not bootable
Media Type:          Generic
Protocol:            ATA
SMART Status:        Verified

Total Size:          149.1 GB
Free Space:          0.0 B

Read Only:           No
Ejectable:           No
OS 9 Drivers:        Yes
Low Level Format:    Not Supported
Device Location:     "A (upper)"
```

ここに表示されたのは，/dev/disk1 のディスク情報であり，総容量，空き容量などが表示されている。

また，**diskutil info /dev/disk0s3** を実行すると，/dev/disk0 の第3パーティションの情報を得ることができる。

```
Device Node:         /dev/disk0s3
Device Identifier:   disk0s3
Mount Point:         /Volumes/Second
Volume Name:         Second

File System:         Journaled HFS+
                     Journal size 24576 k at offset 0x749000
```

```
Owners:              Enabled
Partition Type:      Apple_HFS
Bootable:            Is bootable
Media Type:          Generic
Protocol:            ATA
SMART Status:        Verified
UUID:                00ECF16A-5DA2-3D93-A342-BCC8EF04B384

Total Size:          232.8 GB
Free Space:          158.9 GB

Read Only:           No
Ejectable:           No
Device Location:     "B (lower)"
```

ここで利用した「デバイス名」だが，マウントされているディスクのデバイス名を取得するには df コマンドを利用すればよい。実際 **df -lkh** を実行すると

```
Filesystem     Size   Used Avail Capacity Mounted on
/dev/disk1s9   149G   112G    37G    75%    /
/dev/disk0s3   233G    69G   164G    30%    /Volumes/Second
/dev/disk2s3   153G   137G    16G    90%    /Volumes/Data
```

という結果を得る。この結果から，起動ディスク（マウントポイントが/のデバイス）のデバイス名は /dev/disk1s9 であることがわかる。

★ディスクの検証

diskutil verifyVolume /dev/disk0s3 を実行すると，/dev/disk0s3 のディスクの検証ができる。

```
Started verify/repair on volume disk0s3 Second
Checking HFS Plus volume.
Checking Extents Overflow file.
Checking Catalog file.
Checking Catalog hierarchy.
Checking Extended Attributes file.
Checking volume bitmap.
Checking volume information.
The volume Second appears to be OK.
Mounting Disk
Verify/repair finished on volume disk0s3 Second
```

このように「ディスクユーティリティ」を用いたディスクの操作は，すべて diskutil を利用して行うことができる。diskutil の機能のうち最も有用なものは「ディスクのクリア」である。これは，ディスクの再フォーマット後にディスク全体にランダム（またはすべて0）のデータを書き込む操作であるが，数百 MB のディスク全体のデータ書き込みは長時間が必要となる。このとき「ディスクユーティリティ」を利用してしまうとログアウトができなくなるが，この diskutil を利用することにより，「バックグラウンドでジョブを実行」[6] すれば，ログアウトを行ってもコマンド実行は継続されているので，長時間にわたるジョブの実行が可能になる。

6　コマンドの末尾に & をつけて実行する。

なお, XServe などのオプションとして販売されている「ハードウェア RAID カード」の設定は, この diskutil ではなく megaraid コマンドで行う必要がある。

18.1.2　ディスクイメージの操作

hdiutil を用いると，以下のようなディスクイメージの操作を行うことができる。

- ディスクイメージの作成
- ディスクイメージのマウント・アンマウント・検証
- ディスクイメージのフォーマット変換
- ディスクイメージの光学ディスクへの書き込み

【使用例】

★ディスクまたはフォルダのディスクイメージの作成

hdiutil create -srcfolder directory target.dmg

または

hdiutil create -srcdevice /dev/disk0s3 target.dmg

を実行する。前者はフォルダ（ フォルダ名 directory）からディスクイメージを作成する例であり，後者はデバイス名 /dev/disk0s3 を持つディスクからディスクイメージを作成する例である。ともに target.dmg は作成するディスクイメージの名称である。

なお，作成されるディスクイメージのフォーマットは，デフォルトは UDZO（UDIF zlibcompressed）であり，読み取り専用のディスクイメージが作成される。

★空のディスクイメージの作成

hdiutil create -size 1m target.dmg

または

hdiutil create -size 1m -type SPARSE target.dmg

を実行する。いずれの場合にも -size オプションでディスクイメージのサイズを指定している。前者はディスクイメージサイズが 1M バイトのものを作成しているが，後者は 1M バイトまで拡張可能なものを作成している。

なお，create オペレーションでは以下のオプションも利用可能である。

- -fs フォーマットを指定する。HFS+ などを指定可能
- -volname ボリューム名を指定する
- -encryption 暗号化されたディスクイメージを作成する。デフォルトの暗号化方式は 128 ビット AFS である

★ディスクイメージのマウント

hdiutil attach target.dmg

を実行する。マウントされたときの名称はディスクイメージのボリューム名となる。

★ディスクイメージのアンマウント

hdiutil detach /dev/disk1s3

を実行する。アンマウントの対象となるディスクイメージはボリューム名ではなく，デバイス名を指定する必要がある。デバイス名を調べるには df コマンドを利用する。

★ディスクイメージのフォーマット変換

このためには

hdiutil convert -format UDRW -o new.dmg old.dmg

を実行する。old.dmg のフォーマットを変換し，new.dmg として保存する。この例で指定している UDRW とは，無圧縮の読み書き可能なフォーマットである。

★ディスクイメージから CD-R などを作成

hdiutil burn source.dmg

を実行する。-noeject オプションを指定した場合には，CD-R 作成後にマウントが実行される。

【より高度な使用例】hdiutil を利用すると，「ホームフォルダ」の内容を定期的にディスクイメージとしてバックアップすることが可能となる。例えばホームフォルダが /Users/naito にあり，そのデータを「２台目のディスク」Second にディスクイメージとして保存するためには，以下のようなシェルスクリプトを作成し，それを実行すればよい。

```
#!/bin/sh -f
TARGET="/Volumes/Second"
/bin/mv -f ${TARGET}/backup.dmg ${TARGET}/backup_old.dmg
/usr/bin/hdiutil create -srcfolder /Users/naito ${TARGET}/backup.dmg
```

さらに，このコマンドを cron を利用して，深夜に定期的に実行するように設定すれば，無人でバックアップの作成が可能になる。

なお，cron を設定するには crontab コマンドを利用して，crontab と呼ばれる定時実行のためのデータファイルを作成すればよい。

18.1.3　データのコピー

ディスクイメージなどにバックアップしたデータをフォルダや新規ディスクにコピーするには ditto を利用する。ditto ではリソースも同時にコピーされる。

【使用例】

★フォルダ間のコピー

ditto srcfolder dstfolder

とすれば，srcfolder の中身を dstfolder にコピーできる。これを応用すればディスクイメージからのデータ復元が可能になる。

★フォルダなどのデータのアーカイブ

ditto -c srcfolder dstfile

とすれば，srcfolder の中身を CPIO フォーマットで単一のファイル dstfile にコピーできる。

★アーカイブからのデータの抽出

ditto -x srcfile dstfolder

とすれば，CPIO フォーマットのアーカイブファイルから dstfolder へデータの展開ができる。

なお，pax コマンドを利用することで CPIO フォーマットのアーカイブを作成することも可能であるが，pax コマンドはリソースまでは保存できない。

リソースまで含んでアーカイブを作成したい場合には，フリーウェアの hfspax を利用する方法がある[7]。

【使用例】

★データのアーカイブ

hfspax -w -f foo.pax .

カレントディレクトリを foo.pax にアーカイブする。用いられるアーカイブのフォーマットは，デフォルトでは ustar 形式である[8]。

★アーカイブの展開

hfspax -r -v -f foo.pax

foo.pax をカレントディレクトリに展開する。

★アーカイブの中身の表示

hfspax -v -f foo.pax

foo.pax の内容を表示する。

UNIX のアーカイブ・ダンプコマンドを利用できない理由はリソースフォークがファイルとして認識できない点にあった。そのため，リソースフォークをファイルとして分離することが可能であれば，その後に UNIX のアーカイブコマンドでアーカイブすることが可能である。そのような方法を用いたのが以下の手順である。

ここでは，/tmp/FOO ディレクトリに（リソースフォークを持つファイルの代表例として）Microsoft Word の実行形式をコピーして，それを tar でアーカイブし，さらに，別のディレクトリで展開してみよう。

★リソースを含めてコピーする

CpMac /Applications/MyApps/Microsoft Office X/Microsoft Word/tmp/FOO

これにより，Microsoft Word というファイルが /tmp/FOO に（リソースフォークつきで）コピーされた。この時 ls コマンドで見ることができるファイルの情報は「データフォーク」に関する情報であり，「リソースフォーク」はファイルシステムのカタログ B ツリーに格納されているので，UNIX レベルのコマンドでそれらを見ることはできない。

★リソースを分離する

SplitForks /tmp/FOO

7　hfspax はデフォルトではインストールされていない。

8　-x オプションを用いれば cpio 形式も利用できる。

SplitForks コマンドはリソースフォークをファイルとして抽出するコマンドであり，ディレクトリ名を引数として指定すると，そのディレクトリ以下のすべてのファイルを対象にリソースフォークの抽出を行う。その実行結果として ._Microsoft Word という新しいファイルが作成される。これが Microsoft Word に対応するリソースフォークであり，リソースフォークをファイルとして抽出できたことになる。

★ リソースごとアーカイブする

```
tar cvf FOO.tar ./FOO
```

すでにリソースフォークが FOO 以下にファイルとして存在するので，tar コマンドでリソースフォークまでアーカイブできる。

以上で，/tmp/FOO 以下をリソースフォークを込めてアーカイブできた。逆に，分離されたリソースフォークを「結合する」ためのコマンドは /System/Library/CoreService/FixupResourceForks である。したがって，上で作成した tar アーカイブを展開して FixupResouceForks コマンドでリソースフォークを結合すれば，リソースフォークを込めてファイルアーカイブを完全に展開したことになる。その方法はつぎのとおりである。

★リソースを結合する

```
/System/Library/CoreServices/FixupResourceForks /tmp/FOO
```

この FixupResourceForks は指定されたディレクトリ以下のリソースフォークを結合する。

また，上のようにして CpMac でコピーしたファイルを完全に消去するためには，一旦 SplitForks でリソースフォークを分離して，データフォークとリソースフォークの両方を消去する必要がある [9]。

なお，分離されたリソースフォークのままでも MacOS X の動作には何ら支障はない。

18.2 ユーザ情報の操作

新規ユーザの作成，ユーザの消去などのユーザ情報の操作には NetInfo 関連のコマンド群を利用する。NetInfo の基本的な解説や，LDAP などその他のネーミングサービスに関しては，以前の解説 [4，Section 12] を参照していただきたい。ここでは，NetInfo を利用して新規ユーザの作成などユーザ設定を解説する。

18.2.1 既存のユーザ情報を閲覧する

通常の UNIX システムであれば，ユーザ情報は /etc/passwd ファイルや NIS，LDAP などのネーミングサービスから取り出すことができる。MacOS X ではローカルなユーザ情報は NetInfo データベースに格納されている。初めに既存のユーザ情報を NetInfo データベースから取り出してみよう。

9 どうして RmMac というコマンドが存在しないのかが不思議である。

【ユーザ情報を抽出する】

★**nidump** コマンドを利用する　nidump コマンドは NetInfo データベースの中身を（古典的な UNIX flat DB の形式で）取り出すためのコマンドである。ユーザ情報を nidump を利用して取り出すには，取り出すべきデータベースの名称（ユーザデータベースは passwd）と，対象のドメイン（ローカルであれば“.”）を指定して nidump コマンドを実行すればよい。

nidump passwd .

その結果は

```
nobody:*:-2:-2::0:0:Unprivileged User:/dev/null:/dev/null
root:*:0:0::0:0:System Administrator:/var/root:/bin/tcsh
naito:********:501:20::0:0:Hisashi NAITO:/Users/naito:/bin/tcsh
```

となる。（一部省略してある）

★**niutil** コマンドを利用する　より汎用的な NetInfo データベースの操作コマンドである niutil を利用することもできる。そのためには以下の手順をとる。

▼NetInfo データベース全体の項目リストを取得する　niutil にはドメイン名と「パス名」を指定する必要がある。パス名とは NetInfo データベース上の「位置」のことであり，データベース全体は「ルート」である“/”で指定される。

niutil -list . /

その結果は

```
1         users
2         groups
3         machines
4         networks
5         protocols
6         rpcs
7         services
8         aliases
9         mounts
10        printers
66        locations
```

となる。

▼NetInfo の users データベースの項目リストを取得する　つぎに「ユーザ情報」が格納されている部分 users の項目リストを表示してみよう。今度はパス名として /users を指定する。

niutil -list . /users

その結果は

```
11        nobody
12        root
52        naito
```

となる。（一部省略）

▼個別のユーザの情報を取得する　個別のユーザの情報を取得するには，そのユーザのデータを対象に，-read を実行すればよい。

niutil -read . /users/naito

その結果は

```
picture: /Library/User Pictures/Animals/Butterfly.tif
_shadow_passwd:
hint:
uid: 501
_writers_passwd: naito
realname: Hisashi NAITO
_writers_hint: naito
gid: 20
home: /Users/naito
name: naito
_writers_tim_password: naito
_writers_picture: naito
shell: /bin/tcsh
sharedDir: Public
generateduid: BDF7CE12-0913-11D8-AD5E-000A9599B522
authentication_authority: ;ShadowHash;
passwd: ********
```

上記からわかるように，このデータベースを書き換えることができれば，ユーザ情報を変更したり，新規にユーザを作成，既存のユーザの削除が可能である。

18.2.2　ユーザの作成・削除など

【新規ユーザの作成】

★**niload コマンドを利用する**　niload コマンドは nidump コマンドの「逆操作」を行うものである。新規に“testuser”を作成するには，UNIX flat ファイル形式のユーザ情報を作成しておく。つまり，以下の 1 行が書かれたファイルを作成する。

```
testuser:********:502:20::0:0:TestUser:/Users/testuser:/bin/tcsh
```

このファイルを（例えば）newuser.txt として niload コマンドを実行すればよい。

sudo niload -v -m passwd . < newuser.txt

ここで，新規にユーザを追加するためには管理者権限が必要なため，sudo を経由して実行していることに注意しよう。なお，-m オプションは，既存のデータベースに追加することをあらわす。その結果は

```
1 items read from input
Netinfo /users contains 24 items
Processing input item:
_writers_passwd: testuser
change: 0
class:
expire: 0
gid: 20
```

```
home: /Users/testuser
name: testuser
passwd: ********
realname: TestUser
shell: /bin/tcsh
uid: 502
writing new directory /users/testuser
```

となる。

★**niutil** コマンドを利用する　niutil コマンドで新規ユーザを作成するには以下の手順を行えばよい。

```
niutil -create . /users/naito
niutil -append . uid 501
niutil -append . gid 20
niutil -append . home /Users/naito
niutil -append . shell /bin/tcsh
niutil -append . realname "Hisashi NAITO"
```

これは，新規ユーザの NetInfo エントリを作成し，順にデータを入力している。

【ユーザ情報の変更】ユーザ情報を変更するには niutil コマンドを利用する。

★ログインシェルを変更する　例えば，ユーザ情報のうち，ログインシェルを変更するには

```
sudo niutil -renameprop . /users/testuser shell /bin/bash
```

とすれば testuser のログインシェルが /bin/bash に変更される。

【ユーザの削除】 ユーザを削除するには niutil コマンドを利用するのがよい。

★**testuser** を削除する

```
sudo niutil -destroy . /users/testuser
```

とすればよい。

18.2.3　トラブルの実例

この記事を書いているときに発生した重大なトラブルとその回復状況をレポートしておこう。実は **niload -v -m passwd** を実行するつもりで，誤って **niload -v -d passwd** を実行してしまった。つまり，既存のユーザデータベースを完全に削除してしまった。このとき筆者自身がどのようにこのトラブルから復帰したかを解説しておこう。

このトラブルはユーザデータベースを失ってしまい，ログインも sudo も何もできなくなったことが問題である。そこで，ユーザデータベースを回復するのが一番簡単で，/var/db/netinfo 以下のデータのバックアップが存在する場合には，それを何とかして書き戻せばよい[10]。ところが，そのバックアップは存在していないため，新規に NetInfo データベースを構築する必要がある。

10　この場合には，“Single User Mode”で起動して，データの書き戻しを行い再起動すればよい。

以下はそれを行う手順である[11]。

★ Single User Mode で起動する

この方法は [5，Section 15.6.6] を参照していただきたい。

★ファイルシステムを書き込み可能にする

Single User Mode で起動すると，ファイルシステムは書き込み不可の状態で起動している。それを書き込み可能にするには

```
/sbin/fsck -fy
/sbin/mount -uw /
```

を実行する。

★ NetInfo データベースの再構築と NetInfo の起動

既存の NetInfo データベースをバックアップしてから，データベースの再構築を行う。その後 NetInfo を起動する。

```
/bin/mv /var/db/netinfo/local.nidb /var/db/netinfo/local.nidb.old
/usr/libexec/create_nidb
/usr/sbin/nibindd &
```

を実行する。

★ NetInfo データベースでの情報の書き込み

再構築した NetInfo データベースには，筆者自身のユーザ情報は入っていないので，それを書き込む。

```
/usr/bin/nicl . -create /users/naito
/usr/bin/nicl . -append uid 501
/usr/bin/nicl . -append gid 20
/usr/bin/nicl . -append home /Users/naito
/usr/bin/nicl . -append shell /bin/tcsh
/usr/bin/nicl . -append passwd ""
/usr/bin/nicl . -append realname "Hisashi NAITO"
/usr/bin/nicl . -insert /groups/admin users naito 2
```

を実行する[12]。この段階では「パスワード」は空にしておく。また，「管理者」のリストに自分自身のユーザ名を付け加えておく。

★パスワードの設定

パスワードは

11　ただし，NetInfo データベースには「デフォルトから追加した情報」は「自分自身のユーザ情報だけ」という想定の下の話である。そうでない場合には，「追加されている情報」を復元する必要がある。とりあえず，自分自身がログインできて「管理者権限」が手にはいればよいという状況まで復活させるための方法である。

12　ここで，`-insert /groups/admin users naito 2` の“2”は，この `/groups/admin/users` のリストの 2 番目に naito を付け加えることを意味する。

/usr/bin/passwd

を実行して，インタラクティブに新規パスワードを設定する。

★再起動

これで一連の手順は終了で，データを保存して機器を再起動する。

/bin/sync;/bin/sync;/bin/sync

/sbin/reboot

この後，「アカウント」環境設定で「ログインアイコン」などを設定し直せばよい。

なお，NetInfo データベースの全データを取得するには

nidump -r / .

とすればよい。このデータがあれば niload を使ってデータベースを再構築することが可能になる。

18.3　その他のコマンド

その他の有用なコマンドとして，つぎのものを挙げておこう。

- system_profiler

  システムプロフィールを出力する

- defaults

  OS・アプリケーションなどのデフォルト設定を変更する

- update_prebinding

  システムライブラリなどの情報をアップデートする。例えばアプリケーションの起動が遅くなったときなどに，管理者モードで **update_prebinding -root /** とすると解消される場合がある

- softwareupdate

  ソフトウェアアップデートの実行

【defaults の使用例】

★ユーザ設定情報の取得

defaults read -globalDomain

とすると，ユーザ設定情報を得ることができる。

例えば，**defaults read -globalDomain AppleLanguages** とすると，そのユーザの言語環境設定のリストが出力される。

また，**defaults read com.apple.mail** とすると「Mail」アプリケーションの設定情報を得ることができる。ここで，各アプリケーションに対する「ドメイン名」（この例の com.apple.mail など）は，ライブラリフォルダ内の「Preferenses 」フォルダにある「それらしい名前」から知ることができる。または，**defaults domains** ですべてのドメイン名を出力することもできる。

★ユーザ設定情報の変更　例えば，そのユーザの言語環境設定のリストを変更するには，

defaults write -globalDomain AppleLanguages "(ja, en, fr)"

などとすればよい。この例では，「日本語，英語，フランス語」の順に言語環境の優先順位が設定される。

【softwareupdate の使用例】

★利用可能なソフトウェアアップデート一覧の取得

softwareupdate -l

とすると，利用可能なソフトウェアアップデートの一覧を得ることができる。

```
# softwareupdate -l
Software Update Tool
Copyright 2002-2003 Apple Computer, Inc.

Software Update found the following new or updated software:
   ! iCal155-1.5.5
        iCal, 1.5.5, 7180K [required]
   ! MacOSXUpdateCombo10.3.9-10.3.9
        Mac OS X Update Combined, 10.3.9, 120092K [required]
                                                [restart]
   ! SecurityUpdate
        Security Update 2004-10-27, 1.0, 832K [required]
   * AirPortSW-4.2
        AirPort Software, 4.2, 14652K [restart]
```

この中で！がついているものが，その機器でのアップデート対象のものとなる。

★ソフトウェアアップデートの実行

ソフトウェアアップデートを実行するには，アップデートするものを指定してコマンドを実行する。

softwareupdate -i SecurityUpdate2004-10-27Pan-1.0 iCal155-1.5.5

とすると，指定した 2 つのソフトウェアアップデートを実行することになる[13]。

```
 Software Update Tool
 Copyright 2002-2003 Apple Computer, Inc.

 Security Update: 0...10...20...30...40...
 iCal: 0...10...20...30...40...
 Security Update: 0...10...20...30...40...50...60...70...80...90.
                  ..100
 iCal: 0...10...20...30...40...50...60...70...80...90...100
 Optimizing system performance. This may take a while...
 Done.

 You have installed one or more updates
 that requires that you restart your computer.
 Please restart immediately.
```

リストの中で restart とついているものをインストールした場合には，速やかな再起動を

13　ソフトウェアアップデートの実行には管理者権限が必要である。また，アップデート名はどのような順序で指定してもよい。

求められるので **reboot** コマンドにより再起動を行う。

19　起動の仕組み

最後に MacOS X のシステムの起動手続きを調べておこう。

19.1 起動手続き

多くの UNIX システムの起動手続きは以下のような順序で行われる。（一番最初の部分はかなり省略してある）

1．電源が投入されるとファームウェアが起動し，ハードウェアのテストを行った後に起動デバイスのカーネルやハードウェア拡張を読み込み，ルートファイルシステムをマウントする。
2．init プロセスが起動して，/etc/rc などの起動スクリプトを読み出す。

ここで「起動スクリプトを読み出す」という部分はシステムによってかなり異なる。例えば FreeBSD の場合には /etc/rc を順次実行する中で /etc/rc.conf，/etc/default/rc.conf に記述されたホストごとの設定を評価する。Solaris の場合には「ランレベル」ごとに指定されたディレクトリ（例えば /etc/rc2.d など）の中のスクリプトを，スクリプトの名前の順序にしたがって実行する。いずれにしても，多くの UNIX システムでは各種のサービスを起動する順序は /etc/rc などに記述されていて，起動時に障害が発生した場合には，スクリプトを調べることによりどこで障害が発生しているかが容易に発見できる。

MacOS X の起動手続きは，これらのシステムとは全く異なる方法を取る。その手続きは以下のとおりである。

1．電源が投入されるとファームウェアが起動し，ハードウェアのテストを行った後に起動デバイスのカーネルやハードウェア拡張を読み込み，ルートファイルシステムをマウントする。
2．init プロセスが起動して，/etc/rc などの起動スクリプトを読み出す。/etc/rc では，仮想メモリを有効にした後に SystemStarter とよばれるプログラムを起動する。
3．SystemStarter は StartupItems ディレクトリにある起動スクリプトを，それぞれの依存関係を判断しながら「並行して」起動させていく。

他のシステムの起動手続きと「全く異なる」部分は，起動スクリプトを依存関係を判断しながら並行して起動していく部分である。StartupItems はつぎの２ヵ所にわかれて保存されている。

- /System/Library/StartupItems:
  MacOS X のデフォルトのサービスの起動スクリプト
- /Library/StartupItems
  デフォルト以外の（サードパーティ製などの）サービスの起動スクリプト

以下では /System/Library/StartupItems の内容とその役割を順に見ていこう[14]。

14　/System/Library/StartupItems の内容は Tiger になって大幅に削減され，多くのものが /etc/rc 内で起動されるようになった。

19.1.1 StarupItems の内容

デフォルトで /System/Library/StartupItems に含まれるディレクトリ（起動スクリプト）は以下にあげるものたちである。

- Apache：Apache Web サーバ
  /etc/hostconfig に WEBSERVER=-YES- と記述されていれば apache を起動する
- AppServices：アプリケーションサーバ
  CoreService を起動する
- AppleShare：AppleShare サーバ
  /etc/hostconfig に AFPSERVER=-YES- と記述されていれば AppleShare サーバを起動する
- AuthServer：Authentication サーバ
  /etc/hostconfig に AUTHSERVER=-YES- と記述されていれば tim を起動する
- CrashReporter：クラッシュレポータの起動
  /etc/hostconfig に CRASHREPORTER=-YES- と記述されていれば crashrepoterd を起動する
- Disks：autodiskmount の実行
- FiberChannel：Fiber Channel コントローラの設定を行う
  Fiber Channel カードを搭載する場合のみ関係する
- IFCStart：国際化コンポーネントのキャッシュを再構成する
- IPServices：InternetSharing の実行。/etc/com.apple.named.conf.proxy が存在するときのみ実行される
- Metadata：メタデータサーバ（SpotLight のためのサーバ）の実行
  /etc/hostconfig に SPOTLIGHT=-YES- と記述されていれば lsregister mds を起動する
- NFS：NFS の設定
  /etc/exports が存在すれば NFS サーバを起動し，/etc/hostconfig に AUTOMOUNT=-YES- と記述されていれば Directory Service から NFS ディスクの情報を得て automount を利用して NFS ディスクをマウントする
- NIS：NIS の設定
  /etc/hostconfig に NISDOMAIN が記述されていれば NIS を起動する
- NetworkTime：NTPD を起動する
  /etc/hostconfig に TIMESYNC=-YES- と記述されていれば，ntpd を起動する
- PrintingServices：プリンタサービスを起動する
  /etc/hostconfig に CUPS=-YES- と記述されていれば，cupsd を起動する
- RemoteDesktopAgent：Apple Remote Desktop クライアントの設定
  /etc/hostconfig に ARDAGENT=-YES- と記述されているときのみ

• SNMP：SNMP（Simple Network Managing Protocol）サーバ
/etc/hostconfig に SNMPSERVER=-YES- と記述されていれば snmpd を起動する

これらのことからわかるとおり，多くのサービスは /etc/hostconfig に指定された変数に -YES-，AUTOMATIC+ などの値を与えることにより，サービスの起動を制御できる。

19.1.2 /etc/hostconfig のデフォルト設定

/etc/hostconfig のデフォルト設定は

```
HOSTNAME=-AUTOMATIC-
ROUTER=-AUTOMATIC-
AFPSERVER=-NO-
APPLETALK=en0
AUTHSERVER=-NO-
AUTOMOUNT=-YES-
CONFIGSERVER=-NO-
CUPS=-YES-
IPFORWARDING=-NO-
IPV6=-YES-
MAILSERVER=-NO-
NETBOOTSERVER=-NO-
NETINFOSERVER=-AUTOMATIC-
NISDOMAIN=-NO-
RPCSERVER=-AUTOMATIC-
TIMESYNC=-NO-
QTSSERVER=-NO-
SSHSERVER=-NO-
WEBSERVER=-NO-
SMBSERVER=-NO-
DNSSERVER=-NO-
CRASHREPORTER=-YES-
NFSLOCKS=-AUTOMATIC-
SNMPSERVER=-NO-
SPOTLIGHT=-YES-
ENCRYPINGSWAP=-YES-
```

である。/etc/hostconfig の変数のいくつかは「システム環境設定」を通じて指定される。

- SSHSERVER:「システム環境設定」の「共有」設定で「リモートログイン」を有効にすると -YES- となる
- WEBSERVER:「システム環境設定」の「共有」設定で「パーソナル WEB 共有」を有効にすると -YES- となる
- AFPSERVER:「システム環境設定」の「共有」設定で「パーソナルファイル共有」を有効にすると -YES- となる
- SMBSERVER:「システム環境設定」の「共有」設定で「Windows ファイル共有」を有効にすると -YES- となる
- TIMESYNC:「システム環境設定」の「日付と時刻」設定で「ネットワークタイムサーバ」を有効にすると -YES- となる
- ENCRYPTINGSWAP:「システム環境設定」の「セキュリティ」設定で「安全な仮想メモリを使用」を有効にすると -YES- となる

19.1.3　起動スクリプトの実行順序

これらの起動スクリプトは SystemStarter によって依存関係が判断されると書いたが，それぞれのスクリプトの依存関係は，各スクリプトが保存されているディレクトリ内に記述されている。各起動スクリプトのディレクトリ内には以下のファイル（ディレクトリ）がある。（これは /System/Library/StartupItems/AppleShare の例である）

- AppleShare: 実行される起動スクリプト本体
- StartupParameters.plist: 起動スクリプトの依存関係を記述したファイル
- Resources: 各言語環境ごとに利用される「起動時のメッセージなど」を記述した xml 形式のファイルが格納されているディレクトリ

MacOS X でのスクリプト（アプリケーション）は単体で存在する場合もあるが，このようにディレクトリ全体で一つのスクリプト（アプリケーション）を構成している場合もあり，これを「アプリケーションフォルダ」と呼ぶ。

依存関係を記述したファイル（StartupParameters.plist）は AppleShare の場合，

```
{
  Description     = ``Apple File Service'';
  Provides        = (``Apple File Service'');
  Requires        = (``Disks'', ``DirectoryServices'', ``AppleTalk'');
  Uses            = (``Network Time'');
  OrderPreference = ``None'';
}
```

という内容であり，

- このスクリプトが Apple File Service という「識別子」であり（Provides の部分）
- このスクリプトを実行する際には Disks，DirectoryServices，AppleTalk を必要とし（Requires の部分）
- このスクリプトを実行する際には Network Time がもし存在すれば，その実行を必要とし（Uses の部分）

ことが指定されている。また，OrderPreferences は，同じ属性を持つスクリプトが複数あった場合に，その中での順序を指定する役割を持ち

- None: 特に順序は指定されない
- First: 最も最初に実行する
- Early: 出来る限り早く実行する
- Late: 出来る限り遅く実行する
- Last: 最後に実行する

という属性を指定できる。

したがって“Apple File Service”の実行は Disks，DirectoryServices，AppleTalk の起動が完了した後であり，Network Time が存在している場合には，その起動が完了した後に実行されることがわかる。

19.2　起動スクリプトの追加方法

システムの起動時に実行するサービスを追加するには

/Library/StartupItems ディレクトリにスクリプトを追加すればよい[15]。

起動スクリプトを作成するための手順は以下のとおりである。以下では localconfig という名前の起動スクリプトを作る例を示す。

1．/Library/StartupItems に localconfig ディレクトリを作成する。

2．/Library/StartupItems/localconfig に StartupParameters.plist を作成する。その内容は以下のいずれかである[16]。

```
{
  Description    = "localconfig";
  Provides       = ("localconfig");
  Requires       = ("Network", "Network Time");
  OrderPrefernce = "None";
}
```

```
%<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist SYSTEM
  "file://localhost/System/Library/DTDs/PropertyList.dtd">
<plist version=''0.9''>
<dict>
            <key>Description</key>
            <string>localconfig</string>
            <key>OrderPreference</key>
            <string>None</string>
            <key>Provides</key>
            <array>
                        <string>localconfig</string>
            </array>
            <key>Requires</key>
            <array>
                        <string>Network</string>
                        <string>Network Time</string>
            </array>
</dict>
</plist>
```

前者は「古い形式」のものであり，最新の形式は後者の「XML 形式」を取っているものが多い。

3．/Library/StartupItems/localconfig に実際の起動スクリプトである localconfig を作成する。通常このスクリプトは /bin/sh スクリプトであり，以下のように記述してあるものが多い。

15　MacOS X の基本コンセプトの下では /System ディレクトリ以下はデフォルトのシステムのみが使うことになっているので，サードパーティ製のアプリケーションなどは /Library 以下の対応するディレクトリを利用する。

16　この Requires, OrderPreference は一例であり，スクリプトの内容に依存する。

```
#!/bin/sh
 . /etc/rc.common
 StartService ()
 {
    #### ここにサービス起動のためのコマンド列を記述する。
   ConsoleMessage "Starting Local Configuration"
 }
 StopService ()
 {
   #### ここにサービス終了のためのコマンド列を記述する。
   ConsoleMessage "Stopping Local Configuration"
 }
 RestartService () { StopService; StartService; }
 RunService "$1"
```

スクリプト内で /etc/rc.common を評価しているが，/etc/rc.common 内で /etc/hostconfig が評価されるので，このスクリプト内では /etc/hostconfig で指定した変数を参照可能となる。

このスクリプトは，システム起動時には start，システム終了時には stop を引数として実行され，システム起動時には，スクリプト内の StartService 関数が，システム終了時には，スクリプト内の StopService 関数が実行される。

4．起動時に /etc/rc から呼び出される SystemStarter を利用して起動する場合に，コンソール（起動画面）にメッセージを表示したい場合には，さらに各言語ごとのリソースを作成する必要がある。

そのためには /Library/StartupItems/localconfig 内に Resources ディレクトリを作成し，その中に English.lproj（最低これだけは必要）ディレクトリを作成する。English.lproj ディレクトリ内には Localizable.strings ファイルを以下の形式で作成する。

```
 <?xml version="1.0" encoding="UTF-8"?>
 <!DOCTYPE plist SYSTEM
   "file://localhost/System/Library/DTDs/PropertyList.dtd">
 <plist version=''0.9''>
 <dict>
              <key>Starting Local Congifuration</key>
              <string>Starting Local Configuration Script</string>
              <key>Stopping Local Configuration</key>
              <string>Stopping Local Configuration Script</string>
 </dict>
 </plist>
```

この XML ファイルは，上の localconfig スクリプト内で ConsoleMessage に指定した文字列を key として，ConsoleMessage のかわりに，その key に対応する string をコンソール（起動画面）に表示する意味を持つ。すなわち，localconfig 実行時にコンソールに表示されるメッセージは Starting Local Configuration Script となる[17]。

17　これは，もちろん Aqua 環境のコンソールの話である。

システムが「日本語」モードで動作する場合，Japanese.lproj ディレクトリ内に Localizable.strings が存在すれば，そこに記述された文字列がコンソールに表示される。なお，「日本語文字列」を表示させるためには Localizable.strings の string としては UTF-8 エンコーディングされた文字列を記述する必要がある。

最後に

予定を大幅に超過して，さらには何度も原稿を落しながら，２年以上７回に渡って MacOS X に関する解説を書いてきたが，一応今回を持って終りとしたい。この連載を通じて，MacOS X のより進んだ利用法を理解していただき，安全かつ快適な Mac Life を楽しんで頂ければ幸いである。最後に，この連載を楽しんで頂いた読者の方々に感謝したい。

参考文献

［1］ 内藤久資，Mac OS X －先進的で直感的なオペレーティングシステム－
名古屋大学情報連携基盤センターニュース，2 (2003)，201-245

［2］ 内藤久資，Mac OS X －あなたの Mac は元気ですか？－
名古屋大学情報連携基盤センターニュース，2 (2003)，320-353

［3］ 内藤久資，Mac OS X － Mac OS X の進化論－
名古屋大学情報連携基盤センターニュース，3 (2004)，9-34

［4］ 内藤久資，Mac OS X － Mac OS X のネットワーク－
名古屋大学情報連携基盤センターニュース，3 (2004)，105-149

［5］ 内藤久資，Mac OS X － Mac OS X Server のススメ－
名古屋大学情報連携基盤センターニュース，3 (2004)，290-317

［6］ 内藤久資，Mac OS X －続 Mac OS X の進化論－
名古屋大学情報連携基盤センターニュース，4 (2005)，211-236

［7］ Common UNIX Printing System，**http://www.cups.org/**

［8］ xinetd，**http://www.xinetd.org/**

［9］ The Apache Software Foundation，**http://www.apache.org/**

（ないとう　ひさし：名古屋大学大学院多元数理科学研究科）