Linuxシステム管理入門

2000年11月30日

末安 泰三
日経BP社 日経Linux編集

チュートリアル概要

- □Linuxの基礎を理解する
- □ユーザー管理の基本
- □サーバー運用のため理解したいこと
- □セキュリティを考える

とりあえずできた…では続かない

- □サーバー・ソフトをアップグレードしたい
- □サービスをカスタマイズしたい
- □突然，サーバーが動作しなくなった
- □Linuxが起動しなくなった
- □セキュリティが心配だ…

ディストリビューションの進化で，サーバー・ソフトは簡単にインストールできるようになった。しかし，運用に際しては，Linuxの仕組みについての理解がどうしても必要になる

Linux活用のポイント

- Linux = UNIX
- 枯れた共通技術のみを使う
- 性能に問題あれば特殊機能も利用
- 安定カーネル搭載の機能は利用すると楽
  - ipchains（2.4からはNetFilter）など
- パッケージ管理システムは利用ポリシーを決める
  - 混在はやっかいなエラーを招く恐れがある
  - 異なるディストリビューションのパッケージは使わない
- X Window Systemは使わない

あらためてLinuxとは何か

- □POSIX互換システム
- □GPLを採用したフリーUNIX
- □ソースはすべて公開，改変しての再配布も可能
- □フリーソフトの中心プラットフォーム

他のフリーUNIXに対する利点

□豊富なデバイス・ドライバ
□多数のプラットフォームに対応
□企業の正式なサポート
□ハイエンドでの稼働実績も増加

他のフリーUNIXに対する欠点

□ライセンスにGPLを採用（留保条件あり）

→ 考え方次第では問題にならない

重要なのは使い分け

- □Linuxの主戦場は中規模サーバーまで
  - ○サポートある場合はハイエンド用にも使える
  - ○無理してまで使うメリットは少ない
- □インターネット・サーバーで実績
  - ○負荷高い場合はクラスタリング構成に
- □マルチメディア・デバイスに強い
  - ○DVD-Videoなども再生可能
- □安全性より性能重視の面あり
  - ○ファイルシステムの同期書き込み不完全

LinuxはUNIX

- Linuxの基本的な仕組みは他のUNIX系OSと変わらない
- 知識やノウハウを生かせる
- UNIXは歴史があり，設定ファイルの書式等が不統一
- 一見複雑に見えるが，システムそのものは単純
- 各種設定ファイルは基本的にテキスト・ファイル
- GUI管理ツールは便利だが，応用がきかない

UNIXとの違い

- □カーネル2.4では機能差が少なくなった
- □非同期I/Oの欠如
  - ○スレッドを利用して性能低下回避
- □カーネルレベル・スレッドの採用
  - ○Linuxではプロセスとスレッドは同じ扱い
  - ○__clone()システムコール
    - ▹fork()と似た仕組みでスレッドを生成
- □ツール類はGNUのツールを使用
  - ○便利だが，オプションなどに違い

Linuxにおけるスレッド

- Linuxはカーネルレベル・スレッド
  - プロセス管理の仕組み上，性能悪化は少ない
  - マルチプロセッサ環境できちんと振り分け
- Pthread互換のLinuxThreadsが標準
  - glibcに組み込み可能，もっとも使われている
  - 未実装のメソッドもある
  - 最新版を使うのが望ましい
- BSDで良く使われるPTL
  - 日本人が開発
  - 最近のLinuxでは使用できない

カーネルは安定版を使う

- □最新安定版は2.4.xだが，2001年中は2.2系を使う
  - ○2.4.1までにはファイルシステム破壊のバグあり
  - ○最新カーネル2.4.10にもさまざまな変更
- □2.4を使う場合も，実験採用の機能は使わない
  - ○バグはもちろん，APIなどが変更になることもある
  - ○使用する場合は事前にリサーチして問題洗い出し

Linuxの基本ディレクトリ構成

/ ルート・ディレクトリ
/bin 必須コマンド・バイナリ
(cat, chgrp, chmod, chown, cp, date, ddなど)
/boot ブート・ローダー関連ファイル
/dev デバイス・ファイル
/etc ホスト固有の設定ファイル
(fstab, passwd, exports, networksなど)
/etc/X11 X Window System用設定
/etc/opt 追加アプリケーション用設定ファイル
/home ユーザーのホーム・ディレクトリ
/lib システムに必須の共有ライブラリ
/lib/modules ロード可能なカーネル・モジュール
/mnt 一時マウント用マウント・ポイント
/opt 追加アプリケーション・パッケージのインストール先
/root rootユーザー用のホーム・ディレクトリ
/sbin システム用バイナリ(init, reboot, fdisk, ifconfigなど)
/tmp 一時退避ファイル
/usr マシン間で共有可能なバイナリやデータ
/var ログやスプールなどの可変データ

FSSTNDとFHS

- □Linuxのディレクトリ構成に関する2つの標準
- □Linux Filesystem Structure Standard (FSSTND)
- □Filesystem Hierarcy Standard (FHS)
- □FHSは，Linux以外のOSとの相互運用に配慮
- □現在の最新版は，FHS 2.2
- □ディストリビューションは，FHSへ移行中
  - ○http://www.pathname.com/fhs/

Linuxの設定ファイルはテキスト形式

利点

- □通常のエディタで編集可能
- □障害復旧が容易
- □可読性や検索性が高い

欠点

- □データの表現形式が不統一
- □アプリ間連携が取りにくい
- □記号のら列にしか見えない設定ファイルもある

Linuxのテキスト・エディタ

- 2大テキスト・エディタ「vi」と「Emacs」
- 設定ファイルの編集にはviが向くと言われる
- viは軽量で高機能。すべてのUNIXで利用できる
- Emacsでは，バックアップ・ファイルに注意
  - 作らないようにも設定できる
- 緊急時に利用できるその他のソフト
  - echoコマンド
  - ライン・エディタ「ed」

Linuxの起動シーケンス

- □LILOによるカーネルの読み込みと実行
- □カーネルによるデバイス初期化
- □/sbin/initプログラムの実行
- □起動スクリプトの実行 (各種サービスのスタート)
- □ユーザーへのログイン手段 (gettyやxdm) の提供
- □ユーザーのログイン
- □シェルの起動

カーネルとLILOの関係

- LILOは予めブート・デバイスに書き込まれている
- 書き込みは/sbin/liloコマンド
- 各種設定は/etc/lilo.confファイルで行う
- カーネルの位置情報などを記録
- 起動時には/etc/lilo.confの内容は一切参照しない

そのため，カーネル更新時や/etc/lilo.confを書き換えた際には，
必ず/sbin/liloコマンドを再度実行する必要がある

/sbin/initによる初期設定

- □initは，UNIX系OSで最初に実行される
  - ○システムの初期設定を行う
  - ○全プロセスの親となる
  - ○設定ファイルは/etc/inittab
- □この動作はUNIX系OSで共通

Linuxでは「sysvinit」を利用している

- □「ラン・レベル」により動作が変化

動作モードを表すラン・レベル

- □ラン・レベルはLinuxの「動作モード」
- □0から6の数値，あるいは「s」や「S」の文字で表す
  - ○「s」と「S」はシングル・ユーザー・モード
  - ○管理者1人しかログインできないモード
  - ○重要なシステム変更の際に使用する
- □他のラン・レベルの意味は，システムにより異なる
- □レベル切り替えにはtelinitコマンドを使用する

ラン・レベルとrcスクリプトの対応

- /etc/inittabでは，ラン・レベル別に設定
- ラン・レベルに応じて各種サーバーを起動する
- これにより，各ラン・レベルでの動作が変わる
- サーバーは，/etc/init.d以下のスクリプトで起動
- スクリプト群は，サーバー・ソフトの起動，あるいは停止を行う

初期化スクリプトの管理

- □初期化スクリプトはシェル・スクリプト
- □理解できれば自由に書き換えて構わないが…
  - ○管理を専用コマンドで行うので推奨しない
  - ○書き換えるのは，rc.local程度にする

スクリプト管理用コマンド

- □RedHat Linux系の場合は「chkconfig」
- □Debian GNU/Linux系の場合は「update-rc.d」

例) /etc/rc.d/init.d/pcmciaスクリプトを有効にする

```
# /sbin/chkconfig --add pcmcia
```

管理者の初期化スクリプト編集

- Linux起動時にプログラムを自動起動したい場合
  - rc.localの末尾に起動用のコードを書く
- 記述はシェル・スクリプトの書式で行う
- Linuxの場合はbash用の記述で問題ない
- bash-2.x専用のコマンド(disownなど)は利用しない
- rc.local以外のスクリプトは変更しない方が無難
- rc.sysinitファイルは危険
- 不適切な編集でシステムが起動しなくなることも

起動に問題が生じたときは

□対処法を知っておけば，ほとんどの場合，復旧可能

○安易な再インストールはやめる

□初期化スクリプトに問題がある場合

```
lilo: linux init=/bin/sh
mount -rw -o remount /
```

カーネルの不具合で上記のようにしても駄目な場合は，

ramfloppyなどの緊急用ディストリビューションが役に立つ

サーバーの実体はデーモン

- □デーモン(daemon)はバックグラウンドで動作
- □処理を終えても終了せずに常駐し続ける
- □多くは起動スクリプトなどから起動される
- □特に注意を払わねば，思わぬ不具合を招く
- □スーパーデーモンが起動するデーモンには要注意
  - ○psコマンドで動作が確認できなくても有効のことが

スーパーデーモン「inetd」

- □デーモンを起動するデーモン
- □必要になった場合にだけデーモンを起動
  - ○システム・リソースの消費を回避
  - ○ネットワーク系の処理によく利用される
- □/etc/inetd.confで設定する
- □不必要なサービスは無効にしておく
- □高負荷サービスには向かない
  - ○xinetdなどの代替物あり
  - ○FreeBSDでは改良inetdを採用

/etc/inetd.confでの設定

- 「TCP_Wrapper」との併用が多い
- 無効にしたいサービスは，設定行の先頭に「#」を挿入する
- 設定変更後は，必ずinetdデーモンを再起動する
  - HUPシグナルを送る
  - kill -HUP プロセス番号
  - あるいは killall -HUP inetd
- 適切に運用すれば有用だが，目が行き届きにくい
  - 初心者のうちは完全に停止しておくのも手
  - 頻繁に利用しないソフトは不要とも見なせる

TCP_Wrapperでのアクセス制御

- □ホスト名やIPアドレスに基いたアクセス制限
- □inetdからTCP_Wrapperを介してデーモン起動
- □アクセスを許可するホスト名を/etc/hosts.allowに
- □アクセスを許可しないホスト名を/etc/hosts.denyに
- □基本的にすべてのアクセスは禁止するようにする

RPC(Remote Procedure Call)関連サービスのportmapperでも，

上記の設定ファイルを利用することに注意

採用が増えるxinetd

- □inetdの欠点を克服
  - ○高負荷時にも安定動作できる仕組み
  - ○アクセス制御機能を内蔵
  - ○自動設定ツールに対応できる設定
- □inetdとはまったく設定が異なる
  - ○/etc/xinetd.confが設定ファイル
  - ○/etc/xinetd.d以下にサーバーごとの設定
  - ○「disable=no」としなければ起動しない
- □再起動方法もinetdとは違う
  - ○HUPシグナルではなくUSR1シグナルを送る
  - ○killall -USR1 xinetd

ユーザー管理の基本

□/etc/passwdファイルがユーザー管理用データベース

□パスワードのほかに多くの情報を記録

- ○ユーザー名やユーザーID
- ○ログイン・シェル
- ○ホーム・ディレクトリなど

/etc/passwdファイルの各フィールドの意味

ユーザー名：パスワードのハッシュ値：ユーザーID：
グループID：フルネーム：ホーム・ディレクトリ：
ログイン・シェル

ユーザーIDでリソースを管理

- □Linuxがユーザー管理に用いるのは「ユーザーID」
  - ○ファイル所有者などもユーザーIDやグループIDで管理
  - ○「ls -ln」コマンドでそれを知ることができる
- □ユーザーIDが0のユーザーは「スーパーユーザー」

スーパーユーザー

- □管理用の特権アカウント
- □すべてのファイルやコマンドに対して無制限のアクセス権を持つ
- □ユーザー名は伝統的に「root」
- □root権限の多用は避ける

/etc/passwd操作用のコマンド

- □ユーザーの追加や削除は「useradd」，「userdel」
- □パスワードの有効期限設定には「chage」
- □/etc/passwdファイル編集用の「vipw」
- □パスワードを変更する「passwd」
- □バッチ処理的に複数の情報を更新する「chpasswd」

エディタで直接ファイルを編集するのは避けた方が良い。
マルチユーザー環境では，同じタイミングで更新作業を行っている
可能性もあるからだ

パスワードは暗号化されない

- □パスワードは一方向関数で処理される
  - ○/etc/passwdの情報からパスワード生成は不可能
  - ○単純なパスワードは，「辞書攻撃」で破られる
- □MD5を使うと長いパスワードが利用でき，安全性が高まる
- □Cracklibの導入による安全策
  - ○危険なパスワードへの警告
- □シャドウ・パスワードへの移行
- □telnetやFTPを使用しないようにする

共有ライブラリの管理

- □プログラム実行時に動的にロード(リンク)される
- □リソースの有効活用，ライブラリの更新が容易
- □バージョン管理やパーティショニングがやや難しくなる
- □sonameによる管理
- □Cライブラリのリンクを消してしまったら？
- □基本コマンドだけでも静的リンクした方が良いという意見も
- □トロイの木馬の危険性

共有ライブラリに関するソフト

- □プログラムのリンク状況を調べるlddコマンド
- □共有ライブラリの管理コマンド「ldconfig」
  - ○ライブラリに適切なシンボリック・リンクを作成
  - ○キャッシュを作成

共有ライブラリのリンクの仕組み

□動的ローダー(リンカー)「ld-linux.so」がリンク

□処理高速化のため，キャッシュ(/etc/ld.so.cache)を利用

○指定したディレクトリからライブラリ検索

○ディレクトリは/etc/ld.so.confで指定する

ライブラリの読み込み順

1. 環境変数LD_PRELOADで指定したライブラリ
2. 環境変数LD_LIBRARY_PATHで指定したパスのライブラリ
3. /etc/ld.so.cacheにキャッシュしたライブラリ
4. /usr/libにあるライブラリ
5. /libにあるライブラリ

カーネル・モジュールの管理

- □モノリシック・カーネルの欠点を解消
- □リソースの有効活用，競合する設定などが可能に
  - ○DVD-ROMとCD-Rの場合など
- □ベンダーの一部もドライバをモジュールの形で提供
- □カーネル空間で動作するため，管理には注意が必要

モジュールのインストール先

- □カーネル設定でモジュール化を選択できる
- □インストール先は，/lib/modules/バージョン番号/
- □安全なモジュールだけを使うようカーネル設定可能
- □モジュールのコンパイルとインストール方法

モジュールの利用方法

- □モジュールのロードは/sbin/insmod
- □モジュールのアンロードは/sbin/rmmod
- □ロードされたモジュールの一覧は/sbin/lsmod
- □必要なモジュールを自動的にインストールするkmod
- □カーネル2.2より前はkerneld
- □モジュールには依存関係があることに注意

モジュールの依存関係

- /sbin/depmodコマンドで依存関係のデータベース作成
- データベースによりモジュールを動的ロードできる
  - コマンドは/sbin/modprobe
  - modprobe -rではアンロード

どのような環境でLinuxを活用するのか

- □Linux単体利用
- □複数のLinuxマシンを統合管理（NFSなし）
- □複数のLinuxマシンを統合管理（NFSあり）
- □他のUNIX系OSと統合管理
- □他のUNIX系OS，Windowsと統合管理
- □状況に応じて最適な管理手法は当然異なる

Linux単体利用の際の管理

- □ディストリビューション標準の管理ツールの活用
- □パッケージ管理システムが利用できる
- □ライブラリの管理はldconfigを利用できる
- □各種GUI管理ツールも利用可
  - ○HDE Linux Controller Standard Edition
  - ○linuxconf
  - ○Webmin
  - ○LINSE

パッケージ管理システムの利用ポイント

- □Linuxはパッケージ管理システムが発達
  - ○rpm，debの2大パッケージ
  - ○パッケージ情報はDBに格納
  - ○rpmは対応ディストリビューション多い
- □管理コスト下がるため積極活用したい
  - ○各種GUI管理ツールでも利用が前提
  - ○セキュリティ対策パッケージの迅速な適用
  - ○コンパイラ不要に
- □混用は避ける
  - ○他ディストリビューションのパッケージは使わない
  - ○パッケージを使う場合は，すべてを管理システムで管理

複数のLinuxマシンを統合管理

- □個々のマシンをどのように効率管理するかが重要
- □NFSでのディレクトリ共有は難しい
- □GUI管理ツールによる統合管理
  - ○Caldera Volution
  - ○HDE Linux Controller Enterprise Edition
- □GUI管理ツールのメリット
  - ○作業ミスの低減
  - ○管理ポリシーの一貫化が容易

NFSによるディレクトリ共有の難しさ

- □パッケージ管理システムが共有を想定していない
  - ○マシンごとのパッケージDB
  - ○カーネル，Cライブラリまでパッケージ管理
- □共有ライブラリのリンクにldconfigを使えない
  - ○マシンごとにキャッシュを作成する必要あり
- □ライブラリのバージョン管理が難しい
  - ○単純に上書きアップデートできない
  - ○バイナリ作成時から工夫する必要がある

NFS利用時の共有ライブラリ管理

- □ライブラリはバージョンごとにディレクトリ分け
  - ○gtk+など互換性軽視のライブラリ多い
  - ○sonameによる利点は，NFS利用時は逆にトラブルの素
  - ○頭が痛いのは***-configプログラムを使うライブラリ
    - ▹-R や -rpath オプションを付けるよう書き換え
    - ▹configureスクリプトなどでオプション指定
- □ライブラリ・パスを明示してバイナリ作成
  - ○ldconfigに依存せずにリンク可能にする
  - ○具体的には -R や -rpath オプションを活用
- □もちろん，マウント位置は共通でなければならない

他のUNIX系OSと統合管理

- □認証データの共通化には依然NISが便利
  - ○パスワードのハッシュ方法に注意
  - ○最近のLinuxやFreeBSDはMD5を採用するものが多い
- □NFSによる共用はデータだけならば問題ない
  - ○NFSのバージョン差に気を付ける
  - ○最大ファイル・サイズ等に違い
- □ホーム・ディレクトリの共有には注意
  - ○SSHの鍵など重要なファイルが存在している

他のUNIX系OS，Windowsと統合管理

- □認証データの共通化はNISが基本
  - ○ディレクトリでの管理は互換性に問題あり
  - ○「Windows Services for UNIX」のNISサーバーの利用
  - ○NISサーバーを使わなくてもパスワード同期可能
- □Windows Services for UNIXの注意点
  - ○パスワードの双方向同期はバージョン2.0から対応
  - ○NISサーバーはUNIXのスレーブになれない

共同管理の問題点

- □管理者権限が必要な範囲が広すぎる
  - ○Linuxに限らずUNIXの大きな欠点
  - ○複数環境を統合管理している場合はさらに問題
- □コンピュータ・アソシエイツの「eTrust」
  - ○管理者権限を分割した独自ポリシーを設定可能
  - ○システムコール・テーブルを書き換え，ポリシー適用
  - ○suを使っても，元のユーザー権限で管理できる
  - ○スタック・オーバーフロー対策機能もあり

セキュリティの確保

- □ネットワーク・アタックの2つの型
- □「不正侵入」．「サービス停止(DoS)攻撃」
- □特に被害が拡散する「不正侵入」には要注意
- □サービス停止攻撃も手段が悪質化
- □基本はファイアウオールでの防御
- □ソシアル・エンジニアリングや内部犯行にも注意

不必要なサービスの停止

- □不必要サービスはセキュリティ・ホールの温床
- □安全なサービスを導入し，危険なサービスを停止
  - ○例えば，ssh導入で，telnetやFTPは停止できる
- □サービス停止には起動スクリプトを変更する
- □inetd関連サービスには注意
  - ○inetdそのものも停止するのも良い
- □不要なプログラムもインストールしない
  - ○setuidプログラムの危険
  - ○サーバーにはX Window Systemも不必要

管理者権限の利用は控える

- □「何でもできる」は危険なことであると認識する
  - ○システムの破壊も自由に行える
  - ○安全とは不自由なこと
- □トロイの木馬への警戒
  - ○環境変数PATHに要注意
- □suコマンドやsudoコマンドの活用
  - ○rootで作業する時間を減らす
- □将来的にはPOSIX capabilityが利用できる
  - ○root権限のうち必要な機能のみを分割委譲できる

必ず導入したいssh

- フィンランドSSH Communications Security社が開発
  - ログイン，ファイル転送を安全に実行
  - ポート・フォワーディング機能も持つ
- rコマンドを完全に置き換えることができる
  - telnetやFTPも不要に
- フリーな実装も登場し，利用が容易に
  - OpenBSDの開発チームが作成した「OpenSSH」
  - ディストリビューションも標準採用

バッファ・オーバーフローの危険

- □バッファ・オーバーフロー(オーバーラン)とは何か
- □固定長バッファに，それを超える量のデータを入れた際に発生
- □悪用するとroot権限を奪取される恐れがある
- □Cの標準関数に危険なものが多い
- □最も一般的で，危険性が高いセキュリティ・ホール
- □プログラムのバグのため，基本は対策版へのアップグレード

バッファ・オーバーフローへの対策

- □対策ライブラリの導入
  - ○危険な標準関数を置き換えるライブラリ
  - ○libsafe
    - ▹http://www.avayalabs.com/project/libsafe/
- □特別なCコンパイラで全プログラムを再コンパイル
  - ○StackGuard
    - ▹http://immunix.org/stackguard.html
  - ○stack-smashing protector
    - ▹http://www.trl.ibm.com/projects/security/ssp/
- □対策カーネルの利用
  - ○Openwallプロジェクト
    - ▹http://www.openwall.com/linux/

知っておきたいchroot

- □公開サーバーは，chroot環境で動作させると安全
  - ○サーバー実行に最低限必要な閉鎖環境を作成
  - ○侵入された場合でも被害の拡散を防止できる
- □chrootで，新たなルート・ディレクトリを指定
  - ○起動したいコマンドも指定できる

まとめ

- □サーバー管理は「手順」を理解するだけではだめ
- □Linuxの基礎を知らねば，安全な運用は不可能
- □Linuxは単純なシステムである
- □知識を得るため，設定ファイルは直接編集する
- □理解した上でならば，GUI管理ツールは有効