POPS:FreeBSD における統一された利用者環境の構築

木本 雅彦[1]　　大野 浩之[2]
東京工業大学大学院 情報理工学研究科[1]
独立行政法人 通信総合研究所 情報通信部門 非常時通信グループ[2]

概要

通常 FreeBSD では、利用者が自分の好みにあったアプリケーションを手動でインストールしているが、これはまったくの初心者にとっては困難な作業であり、熟練者が複数台のホストを立ち上げる場合などは煩わしい作業である。著者らは管理を容易にするために二つのコンセプトを導入した。共通の環境を提供することと、ホスト固有情報と共有情報を分離することである。前者のために、FreeBSD 用のパッケージ集である POPS を開発し公開、配布した。また後者を実現するために FreeBSD の起動処理を改良した。本論文では、POPS の概要およびその実装、起動処理の改良について述べ、これを導入する利点を主にシステム管理者の視点から評価する。

POPS: Construction of common environment on FreeBSD

Masahiko KIMOTO [1]　　Hiroyuki OHNO [2]
Graduateschool of Information Science and Engineering,Tokyo Institute of Technology. [1]
Emergency Communications Group, Information and Network Systems Division,
Communications Research Laboratory. [2]

ABSTRACT

In the FreeBSD world, administrators should install as many packages as they need by their own hand. This is not only hard for beginners, but also for experts: for the former it is impossible to cover all the existing softwares, and even for the latter, it is very bother to choose the appropriate software for their own purposes and install them as many machines as they have. We have introduced two concept to construct an environment on FreeBSD. It resulted that beginners could easily obtain usable environment on FreeBSD, administrators' work of installation was decreased and upgrading became easy. In this paper, we describe POPS and some technologies which realize these concepts.

1　はじめに

Linux の世界では、ディストリビューションをインストールすることによって、利用者は一通り日常的に利用できる環境を構築できる。これに対して FreeBSD の世界では、管理者が必要な多数のパッケージを手動でインストールしていかなければならない。この作業は初心者にとってばかりではなく熟練者にとっても煩わしい作業である。前者については最低限の作業を可能にするためにどのパッケージをインストールすれば良いかという知識が欠如している。後者は逆に作業に慣れてしまい、何度も同じ作業を行なうことが苦痛である。

著者らは FreeBSD における環境構築に、二つのコンセプトを導入した。この結果初心者は簡単に実用に耐える環境を入手でき、熟練した管理者は導入作業やシステム更新作業の軽減といった恩恵を受けられるようになる。本論文では、これらのコンセプトを実現する POPS[1] とその周辺技術について述べる。

2　背景

著者らは 1995 年から PICKLES SYSTEM を開発してきた [2]。これは現在では Linux で言うところディストリビューションに相当するという表現が端的であり、BSD/OS をベースに開発していた。PICKLES SYSTEM は著者らが東工大で運営していた研究室のサーバおよびクライアントとして運用されていた。PICKLES SYSTEM の主要な技術的特徴は以下の二点である。

1. 共通の利用者環境の提供。
2. ホストに依存した情報と全ホストで共有でき

る情報との明確な分離。

本論文では PICKLES SYSTEM で培われたこれら二つのコンセプトの、FreeBSD への適用について述べる。以下では 2002 年 5 月時点での最新リリースである FreeBSD 4.5-RELEASE を対象とする。

コンセプト 1 を実現するために、著者らは POPS を開発した。POPS は端的に述べれば、FreeBSD 用のパッケージ集であり、開発の目的は以下であった。

A 初心者向けに参考になる環境を提供する。

B 多数のホストに対して同じ環境をインストールする場合の工程を容易にする。

C それなりの強度のセキュリティを確保する。

A については、UNIX では口頭伝承の文化が強く、昔は大学や会社で先人が作りあげた計算機環境に触れることで初心者は勉強したものだが、昨今ではそういった環境が周囲にない利用者も多い。このような利用者に「これだけ揃っていればとりあえず利用できる」という参照となる環境を提供することが目的の一つである。

B については、多数のホストに同じ環境を導入する工程を容易にする必要がある。すべてのホストがネットワークで密に連結されている場合などは、ネットワーク経由での同時インストールやディレクトリ自体をネットワークで共有してしまう方法がある。しかし複数箇所に散在している機器やノートパソコンのように、散逸的に導入作業が発生する場合は、例えば CD-ROM にすべてをまとめておくというようにネットワークに依存しない方法の方が適している。

C のセキュリティについては、理想的には全てのアプリケーションが常に最新のものに追従されているべきであるが、現実的には全く保守されていないいわゆる放置サーバが多数存在し、攻撃の対象になっている。2001 年初頭に連続して発生した公官庁の WEB サーバの乗っ取り事件も、被害にあったのはこういった放置サーバであり、それなりのセキュリティが確保されてさえいれば回避できた問題であった。そこで、理想的に完全ではないもののそれなりの安全性が保証されたリファレンスを提供することも目的の一つとする。

コンセプト 2 を実現することによって、システムの更新や復旧処理、バックアップ作業などが容易に行なえる。これは、著者らが PICKLES SYSTEM の開発を通じて示して来た点である [3] 。

3 設計

3.1 FreeBSD におけるアプリケーションの導入

最初に FreeBSD におけるアプリケーションのインストール方法について概観しておくことにする。ここで述べるのは、ソースファイルを入手してドキュメントに従ってコンパイルしてインストールといった作業ではなく、それを簡易化するために採用されている機構についてである。

Linux では RPM[4] や APT[5] といったバイナリ形式でのアプリケーションの配布と導入支援機構が容易されている。これに対して FreeBSD では ports と packages という二つの方法がある。どちらもカテゴリ分けされて管理されている。ports はソースファイルをネットワーク経由で入手して必要ならパッチを当ててコンパイルしてインストールする。コンパイルオプションや必要なパッチ当てなどの作業を自動化したものである。packages は ports によってコンパイルされてインストールされたバイナリをアーカイブにまとめたものである。package は pkg_add というコマンドでインストールでき、アーカイブ内にそのプログラムが必要とするライブラリなどについての情報も格納されているため、pkg_add を実行すると依存している他の package も同時にインストールされる。ports と packages は基本的に一対一に対応しており、port からは `make package` を実行することで package を作成できる。

ports の欠点はコンパイルできない状態が発生しうることである。これは ports が対応するバージョンが古すぎて配布サイトからソースコードが消滅してしまっていたり、同じファイル名のソースコードなのに内容が変わってしまってパッチが正常に適用できなかったりといった原因がある。対して package はコンパイル済のバイナリ配布なので、インストールに失敗することはないが、依存するライブラリや他のアプリケーションのバージョンの検査が厳しく、本来問題なく動作する程度の差異しかないのに依存する package の更新も要求されることがある。逆に依存関係が整理されている package の集合であれば、問題なくインストールできることになるため、次節で述べる POPS では packages のみを用いてユーザ環境を構築する。その際、インストールする順番などで問題が生じたが、その問題と解決手段についても言及する。

利用者数	アプリケーションの数
1	534
2	175
3	85
4	50
5	31
6	19
7	15
8	11
9	12
10	9
11	5

表 1: アプリケーションの利用傾向

3.2 POPS の設計

POPS は著者らが利用しているアプリケーションのパッケージを集めたものである。現状では著者らの近隣者の要求のみを考慮したものになっているが、今後類似の要求が発生することを考えるとこの作成手順について述べる必要があると考えられる。

まず最初に、パッケージ間の依存関係について調査した。パッケージ間の依存関係は機械的に記録されているのだが、これに当てはまらない依存関係が存在する。この整理は手動で行なう必要があった。

著者らはまず最初に内輪の研究グループに働きかけて、FreeBSD を日常業務に利用しているユーザが使っているパッケージのリストを収集した。

11 名のユーザからリストを収集しバージョン番号を取り除いてアプリケーションの名前だけに着目して集計した結果、表 1 のようになった。当初予想していたよりも、全員が共通して使っているアプリケーションは少ないといえる。しかしリスト全体を俯瞰すると、アプリケーションとしては同じものであるのにパッケージの構成が変わって名称が変更されたために複数として計上されていないものがある。言い替えると、複数の世代のアプリケーションのリストが混在しているため、重複が少ないものと考えられる。これはつまり、一旦インストールしたアプリケーションは、特に必要がない限り更新されないものが多いとも言えるかもしれない。

次に FreeBSD の配布サイトから、最新のパッケージの一覧を入手し、先のリストと照合して、インストールするべきパッケージの最新のリストを作成した。

次にこれらのパッケージを三種類に分類した。1) 通常通り pkg_add を実行すれば良いもの、2) 他のパッケージを上書きするために pkg_add の順番に留意する必要があるもの、3)pkg_add を-f オプション (強制インストール) 付きで実行するものである。2) については、例えば foo というライブラリと、それを日本語化した ja-foo というライブラリのパッケージが存在した場合、ja-foo は foo を上書きするので ja-foo だけがインストールされていれば foo を利用するアプリケーションも動作するが、依存関係の情報から foo がインストールされていることも要求されていることがある。この場合、最初に foo をインストールしてから、ja-foo をインストールする必要があるので分離した。

3.3 情報の分離の方針

著者らは PICKLES SYSTEM において、ハードディスク上の情報を二つに分類した。すべてのホストで共有できる情報と、ホスト固有の情報である。前者はシステム動作中は書き変わらないはずなので、読み込み専用ファイルとして設定できるはずである。

UNIX ではこの二つの情報が、ディレクトリごとにおおまかに分類されている。しかし、例えば設定ファイルのほとんどは/etc に置かれているが一部は/usr/local/etc 以下や/usr/X11R6/lib/X11 以下に置かれているというように、完全には分離されていない。ホスト間で共有できる部分を/usr/local/ 以下に集約して NFS で共有する管理手法は広く行なわれているが、この情報を完全に分離することはそれほど容易ではない。

まず/etc 以下には設定ファイルが置かれるが、すべてのファイルがホストに依存したファイルということはない。起動スクリプトなども含まれており、このディレクトリは両者が混在している。/var はその名称が示す通り変更されうる情報が格納されており、この下にはホスト固有情報が格納されている。ホームディレクトリなどのその他の情報は、運用方針によるが、著者らはこれを/local/の下に集約することにした。

PICKLES SYSTEM では二種類の情報を明確に分類し分離した。ホスト共有の情報は「システムディスク」に格納し、ホストごとに依存した情報は「ユーザディスク」に格納する。両者は二台のディスクに分けて格納する場合もあるし、一台のディスクに格納する場合もある。上記を整理すると表 2 に

Directory	含まれる内容	分類
/	root partition	systemdisk
/usr	アプリケーション	systemdisk
/etc の一部	設定ファイル	userdisk
/var	可変ファイル	userdisk
/local	ホームなど	userdisk

表 2: 情報の分類

なる。

4 実装

4.1 POPS の実装

POPS はパッケージの集合とインストールスクリプトから構成されている。著者らはこれらを一つの CD-ROM イメージの形式にまとめて配布しており、その大きさは約 581MB である。

POPS をインストールするためには、対応したバージョンの FreeBSD をはじめにインストールしておく必要がある。この時/usr には 2.5GB 以上を割り当てる。あとは POPS のインストールスクリプトを実行するだけで、すべてのインストール作業が終了する。インストールには、Celeron 500MHz と 24 倍速 CD-ROM の組み合わせのシステムで約 1.5 時間程要する。

以下ではインストールスクリプトの動作について述べる。インストールスクリプトは、パッケージを三段階に分けてインストールする。インストール作業中は、pkg_add コマンドが異常終了しても、無視して処理を継続する。これは pkg_add コマンドの終了ステータスが、「依存しているパッケージファイルが存在しない場合」「既に pkg_add されている場合」「パッケージファイルが壊れている場合」を区別しないためである。次に POPS の環境に適したドットファイルのサンプルを、/usr/share/skel/以下にコピーする。最後に/var/db/pkg 以下に pops のバージョン番号を含んだディレクトリを作成する。これは現在インストールされている POPS のバージョンを記録しておくためである。

更に、ダウンロードしたパッケージが改善されていないことを確認するためのスクリプトを用意した。このスクリプトは手元のパッケージファイルの MD5 と、POPS のマスターサイトにある個々のパッケージの MD5 とを比較するものである。

4.2 情報の分離

以下では二種類の情報の分離方法について述べる。説明を簡易にするため、PICKLES SYSTEM で導入したシステムディスクとユーザディスクという用語を用いることにする。両者の定義は既に述べた。

情報の分離を実現するためには、以下の二点が問題になる。

- /etc 以下の一部だけをどのようにユーザディスクに格納するか。
- ユーザディスクのパーティション情報をどのように発見し処理するか。

前者の点については、union ファイルシステムを用いて解決する。まず最初にすべての設定ファイルを/etc 以下に集約する。例えば/usr/local/etc は/etc/usr.local に移動し、/usr/local/etc からシンボリックリンクを張る。次に/u/etc というディレクトリを作成し、このディレクトリを/etc に union mount する。union ファイルシステムは、あるディレクトリを他のディレクトリの上に「かぶせる」ことができるファイルシステムであり、下のディレクトリのファイルを編集するとまずそのコピーが上のディレクトリに作成され変更後のものが記録される。/u は独立したユーザディスクのパーティションに格納する。これによって、/etc 以下のファイルのうち、修正が加えられたものだけがユーザディスクに格納されることになる。

既に述べたように、ユーザディスクはシステムディスクと同じドライブに格納される場合もあるし、異なるドライブに格納されることもある。このため、起動時にユーザディスクが存在するドライブとそのパーティション構成を検出しなければならない。そこで、起動スクリプトに以下の修正を加えた。

- 候補となるパーティションを検索して、/u にマウントすべきパーティションを発見する。
- /u/etc/にある fstab.userdisk の情報を反映させる。このファイルには、ユーザディスクのパーティション情報が書かれている。
- /u/etc を/etc にマウントした後に、改めて/etc/rc.conf の情報を起動スクリプトの環境変数に反映させる。

/u の候補となるパーティションのリストは、/etc/userdiskcandidates というファイルに列挙されている。checkuserdisk というコマンドを作成し、この

ファイルに書かれているパーティションを検査して.userdisk というファイルを発見したらそのパーティションを/u にマウントするようにした。この状態で、ユーザディスクのどのパーティションをどこにマウントすべきかという情報は/u/etc/fstab.userdisk に書かれている。

/etc/fstab は、libc に中に含まれる getfsent() や getfsfile() などの関数を経由してアクセスされる。これらの関数を変更し、/etc/fstab と/etc/fstab.userdisk という両方のファイルの内容を反映するようにした。またシステムディスク側の/etc/fstab.userdisk から/u/etc/fstab.userdisk へのシンボリックリンクを作成した。この結果、/u/etc が/etc に union マウントされていない状態でも、二つの fstab を読みこめるようになった。

以上の変更によって、ユーザディスクを発見し、/u にマウントした状態で mount コマンドを実行するとシステムディスクとユーザディスクの両方が適切にマウントできるようになった。

次に/etc/rc.conf を再度読みこむ。これは/etc の上にマウントされたユーザディスクの rc.conf の内容を反映させるためである。そして rc の残りの起動手順を続行する。

以上が、システムディスクとユーザディスクの分離を実現するために加えた変更点である。

5 考察

5.1 他の機構との比較

POPS は多数のパッケージを矛盾無く自動的にインストールしたものなので、RedHat や Debian といった Linux のディストリビューションと比べると、管理ツールが不足しているなどの見劣りする点が多い。しかしその意義は、FreeBSD に統一した環境を提供することにあると考えており、論点は FreeBSD のコミュニティでこのようなパッケージ集に対する需要があるか否かという点になる。著者らは 2002 年 3 月に、FreeBSD-Users-JP メーリングリストに POPS をアナウンスした。その後の 4 週間で約 500 件のダウンロードがあった。この数からすると、利用者の需要はそれなりのものがあると考えられる。

多数の計算機に同一の環境を構築する機構としては、Linux 用の LUCIE [6] や SUN の JumpStart[7] などがある。どちらも LAN に接続された初期状態の計算機に、ネットワーク経由で自動的にシステムをインストールする機構であり、クラスタを実装する場合などに有用である。しかし POPS とその元になった PICKLES SYSTEM では、導入作業場所自体がネットワーク的に散逸している状況を主に想定しており、対象としている問題空間が若干異なると言える。

5.2 有用性に関する考察

インストール作業の簡易化については、著者の一人が FreeBSD と必要と思われるパッケージを手動で選択してインストールした時にはおよそ 4 時間を必要としたが、POPS を用いる場合は合計で 2 時間程度で完了した。また POPS の場合はインストールスクリプトを最初に実行させて待つだけであり、パッケージをメニューで選択するといった作業を行なう必要がない。

システムディスクとユーザディスクを分離したことにより、理想的には/usr が格納されるパーティションを読み込み専用属性で利用できることになり、これは組み込み用途などで有効であると考えられる。システムディスクとユーザディスクとを異なるディスクに格納した場合は、システムディスク側の故障対策やバージョンアップがディスクの交換だけで可能である。

POPS のようなパッケージ集が有用であると認知された場合、他にも同様のパッケージ集が登場する可能性があるし、ミラーサイトでの配付が行なわれる可能性もある。そうなると、必らずしも信頼できないサイトが実行形式のバイナリ配付を行なうことになり、トロイの木馬の危険性が予想される。POPS については、パッケージの MD5 を配付サイトで管理する物と比較する方法を採用したが、安全性の確保については引き続き検討しなければならない。またパッケージ集の散発は、Linux の世界で多数のディストリビューションの存在がもたらしている渾沌を鑑みると、慎重に考えなければならない。

基本的に POPS は私的パッケージ集であり、その作成過程からして著者らの普段の利用形態に依存している。例えば日本語以外の言語での利用はまったく想定していない。これらの点についても考慮する必要がある。完全に多国語化を目指すか、言語に依存しない部分だけを配付するかといった選択肢がありうるだろう。

5.3 技術的考察

POPS については、理想的には他のパッケージの依存関係の頂点となるメタパッケージとして実装するのが望ましい。しかし POPS でインストール

手順を三段階に分離せざるを得なかった理由から、現状で単一のメタパッケージとして実装するのは難しい。単体のメタパッケージとするためには、依存関係の矛盾を無くし、日本語化されたライブラリなどを元のライブラリに依存するようにする必要があるだろう。または、POPS と同じように三段階に分けてメタパッケージ化する方法もある。いずれにせよ、現状の/var/db/pkg 以下に空ディレクトリを作成する方法は暫定的に過ぎるので改善方法を検討しなければならない。

次に portupgrade との関係について述べる必要がある。portupgrade はインストールされているパッケージを ports や packages を使って最新版に更新するものであり、パッケージの依存関係の追跡なども自動的に行なう。portupgrade を使えばアプリケーションを容易に最新版に保つことができる。しかし例えば ghostscript のようにバージョンによって ports の構造が大幅に変わってしまうものなどには対応できないので、雛型となるパッケージの一覧は必要になると考えられる。POPS などでベースになるパッケージ集をインストールしておいてから、portupgrade で最新版に更新するという使い方が適切であろう。

本文中では、設定情報を/etc 以下に集約するとだけ記述したが、現在の FreeBSD では/usr/local/etc を/etc 以下に移動するだけでは不十分である。例えば X Window System の多くの設定ファイルは/usr/X11R6 以下に置かれたままである。ただし、この点は XFree86 4 系列では改善されている。その他にかな漢字変換サーバの辞書や VFlib の vfontcap も FreeBSD では/usr/local/以下に置かれているため、これらも/var や/etc に実体を移動する必要があるだろう。

起動スクリプトに対する変更については、本文中では FreeBSD 4.5-RELEASE に対する変更点を述べたが、同じコンセプトを他の BSD 系列の OS に適用する場合のことを考える必要がある。というのも、/etc/rc の処理については NetBSD では rcorder という機構 [8] が導入されているが、この機構の FreeBSD への移植が進んでいるからである。rcorder は個々の起動スクリプトの実行順番を最初に決定して順次実行するため、ユーザディスクをマウントしてからその内容を反映させることができない。rcorder を二段階に分けて実行するなどの方法が必要になるであろう。

6 おわりに

本文中で述べたように、二つのコンセプトは管理作業を軽減する上で有用であり、またこれらのコンセプトは他の OS、例えば他の BSD や Linux などへも応用可能である。これらのコンセプトは決して著しく斬新なものではないが、これを完全に実現しているシステムはない。本論文では、PICKLES SYSTEM の開発経験からその有用性が示されているこれらのコンセプトを、どのように FreeBSD に導入するかについて述べた。FreeBSD の世界でこのようなディストリビューション的なアプローチがどの程度認められるかについてはダウンロード件数などを見るに、それなりに利用者の興味は集めているようである。今後の POPS については、パッケージの内容を整理して必要なものをしぼった上で、方向性を定めて肉付けを行なう必要がある。今のところ有望なターゲットとしては教育用プラットフォームの提供などを考えている。

最後に POPS に関する情報は、以下の URL から入手できる。

http://www.ohnolab.org/ kimoto/freebsd/pops.html

参考文献

[1] 木本雅彦, POPS: Package Of the PackageS ～誰でも簡単 マイパッケージ集～, FreeBSD PRESS NO.9, 毎日コミュニケーションズ

[2] 木本雅彦, 大野浩之, 街角公衆情報端末計画 ～PICKLES の概要～, Mar. 1996, 第 52 回全国大会 講演番号 3Y-2

[3] 木本雅彦, 大野浩之, 自律型ネットワーク端末 (PICKLES) を用いたシステム運用技法, Feb. 1998, 情報処理学会,DSM シンポジウム

[4] Donnie Barnes, RPM How/To, http://www.redhat.com/ support/wpapers/rpm-howto.pdf

[5] APT HOWTO, http://www.debian.org/doc/manuals/apt-howto/index.en.html

[6] LUCIE — The Linux Universal Configuration and Installation Engine, http://matsu-www.is.titech.ac.jp/ takamiya/lucie/docs.html

[7] Rob Snevely, JumpStartTM: NIS と sysidcfg, Oct. 1999, http://www.sun.co.jp/blueprints/1099/jumpstart.pdf

[8] The Design and Implementation of the NetBSD rc.d system, http://www.mewburn.net/luke/papers/rc.d.pdf