Linux/Alpha 活用講座

清水 尚彦　nshimizu@keyaki.cc.u-tokai.ac.jp

第21回
完全ディスクレスマシンの構築

はじめに

●意味不明な訳びっくり！

1年ほど前に出版されたオーム社の「Linux 2.0カーネルブック」(記事末RESOURCE[1]を参照)を、ようやく最近読んでいたのですが、ページテーブルエントリ(Page Table Entry)を「入口」と訳していたりするので、読んでいてひっくり返ります。そのほかにも、いたるところに意味不明な訳が(信じられないほど)多く、読んでいてひどく「消耗」する本でした。これなら、英語のままのほうがずっと読みやすいですね。元の本が良くても、まともな訳がなければ役に立たないと実感するためには良い本だと思います。もっとまともな訳者が増えなければ、日本の技術レベルを疑われても仕方がないですね。こういった本を読んで、自分の理解力が足りないなんて謙虚な気持ちで落ち込む人がいないことを願っています。出版社の人も、もう少しセンスをもって内容をチェックして欲しいものです。

●Plamo Linux 2.0をインストール

ゴールデンウィークの最中に、実家の近くでSZLUG([2])とLSWG([3])のセミナーがあるということなので、普段出かけない私も、妹に車で送ってもらって出席しました。セミナーの後の懇親会でPlamo Linux 2.0のCD-ROMをいただいたので、さっそく実家のパソコンにインストールしました。実家のパソコンは、オークションで入手したIBM PC710なのですが、ハードディスクの容量が少し足りないことを除けば、文章の作成には十分な働きをしてくれています。容量が十分あれば、Windowsとの共存をしても良かったのですが、大きなハードディスクはBIOSが認識しないので、諦めてLinuxオンリーにしました。この方が潔くて良いかもしれませんね。Green PCとして知られているPC710の初期モデルは、VRAMが貧弱なのが原因だと思うのですが、AfterStepを立ち上げると、カラーマップの不足で、ろくにアプリケーションが動かないので、ウィンドウマネージャは「qvwm」にしています。PC710は、Linuxとの相性は良く、サスペンドのボタンで、サスペンド／リジュームが気持ち良く動きます。

●再び電源故障……

研究室のサーバ(AlphaStation 200 4/100)の電源が再び故障しました。この電源は、この前壊れて入れ替えたばかりなので、何となく釈然としません。すでに保証が切れてずいぶん経つこのマシンは、3.3Vの出力を持つAT互換電源という「変わりモノ」を搭載しているので、交換するとすれば定価がずいぶん高い部品をサービスから買わなければなりません。そこで、取りあえず同モデルを使っている学生用のクライアントをつぶして、電源だけ載せ換えて急場をしのいでいます。

時間があれば、たいした部品を使っていない電源回路でしょうから、故障個所を調べて部品交換したいのですけどね。これに懲りて、IPテークオーバーができるミラーサーバを作っておけば良いのですが、なかなか面倒で検討もしていません。どこからもログインできない事態を避けるために、NISのスレーブサーバくらいは作っておきたいのですが、マスターサーバ(Digital Unix 3.2c)の設定の問題なのか、スレーブにうまく情報が渡らないので、要検討のまま放置してあります。Linuxをマスターサーバにすればいいはずなので、近々に対応したいと思っています。

●プレゼンテーション用マシン

プレゼンテーション用に入手したノートパソコンの「Armada M300」に、Debian／potatoをインストールしました。Linuxは、Kondaraをインストールしていたのですが、普段使っているものと勝手が違うと不便なので、入れ換えることにしました。最初は、ノートパソコンへのインストールということで心配していましたが、x86のディストリビューションは利用者が多くてこなれているせいか、思ったよりも簡単にインストールできました。x86とAlphaで同じディストリビューションを使うようになると、Alphaでおかしな振る舞いをしたときに、パッケージングの問題なのか、Alpha固有の問題なのか、切り分けができるようになるのでいいですね。主目的のプレゼンテーションツールとして利用をしているMagicPointも、まだ使いこなしはできていませんが、作図に使いなれたTgifが使えるので、重宝しています。

なお、このマシンは、サスペンドボタンがあるのですが、Xを実行中にサスペンドするとおかしくなってしまいます。そこで、コンソールを切り替えてapmコマンドを発行するスクリプトを作り、ウィンドウマネージャのメニューから起動できるようにしています。一般ユーザーからサスペンドを発行するためにsudoコマンドを使いますが、パスワードを無効にするように/etc/sudoersに記述しておかないと、非対話的なスクリプトからは実行できなくなります。スクリプトで使うコマンドを実行するために、/etc/sudoersをリスト1のようにしています。このファイルはvisudoコマンドで編集してください。

●Windows CEマシンの可能性

Windows CEのマシンにLinuxを載せられると、特に電車の中でモバイルギアIIで原稿を書くことの多い私は便利になるので、安定動作するのであればぜひ試してみたいと思っています。しかし、ROMに入っているWindows CEのOSと違って、値段は下がったとはいえ高価で寿命も限られているフラッシュメモリを頻繁にアクセスして使うのはちょっと心配ですね。しかもフラッシュメモリは書きこみ時の消費電力がそれなりに大きいので、電池寿命の面でも心配はつきません。やはりROMにLinuxを入れたポータブルマシンをぜひどこかで作ってほしいものです。ROMに焼くとなると相当安定していないといけないので、変化の早いこの世界では思い切れないところがありますが、使いたい機能はそれほど多くないので割り切りができると思います。

しかし、それにしてもCEマシンは電池寿命面から見ると、性能が悪いですね。これがOSの問題なのかマシンスペックの問題なのか気になるところです。GUIのためなどに、Windows CEは必要以上に電力を浪費するような振る舞いをしているのではないでしょうか？　だとすれば省電力を狙って思いきり割り切ったLinux機を作るのは有意義だと思います。もちろん、MIPS16などの拡張がされていたとしても、元々MIPSアーキテクチャは省電力に向くように設計されているわけではないので、マシンの基本スペックの部分ですでに電力を浪費するようになっているのかもしれません。だとすれば、むしろ狙いはPsionなどのARMマシンの方がいいのかもしれません。こういう話は考えているだけでもわくわくしてくるのは私だけでしょうか？

Anviの修正

●potatoのnviの不具合

私は現有のマシンの多くをDebian/slinkで運用していますが、持っているマシンのうち1台だけDebian/potato環境にしています。ところが、この環境で、Alphaではpotatoのパッケージで通常使うnvi-m17nというエディタがうまく動作せず困っていました。実はslinkのリリースの頃にも、やはり動かないと困ったことがあったのですが、jelvisを個別にインストールすることで対処していました。

今回も同じことができるのですが、どうせなら原因を追求してみようと若干の作業をした結果、とりあえず動作はするようになりました。同様の事態で困っている方もいるかもしれないので簡単に作業の内容を報告します。

●ソースから再構築

まず、nvi-m17nというものは、vi互換のエディタです。パワーのあるマシンならばEmacsのVIPモードで使っても悪くはないのですが、私にはいまいちしっくりこないのでviが手放せません。Debian/potatoのパッケージでは、これがうまく動かなくて苦労していたのですが、ようやく動かすことができました。デバッグしている当人にとっては長い道のりなのですが、分かってしまえば難しくないのでした。まずはapt-getコ

リスト1　sudoコマンドでパスワードを聞かれないための設定

```
nshimizu ALL = NOPASSWD: /usr/bin/chvt *, /usr/bin/apm *
```

マンドでソースパッケージを取り出しましょう。

```
# apt-get source nvi-m17n
```

これで必要なファイルを展開してくれるので、

```
# cd nvi-m17n-*
```

としてディレクトリを移動したあと、「debian/rules」というファイルを変更します。

このファイルの6行目に、configureへのパラメータが書かれていますが、ここに

```
--disable-curses
```

を追加して下さい。これで無事動作するパッケージを作る環境ができます。canna版も同時に作るようなrulesファイルになっているので、canna版が必要ない方は"build-stamp:"のcanna版に関する3行ほどの記述をコメントにすれば良いでしょう。

```
# debian/rules build
```

でバイナリを作った後

```
# build/nvi
```

で試してみてください。以前、slinkのリリースの頃に"--disable-curses"だと文字化けするというような話が出ていたのですが、新しいcursesではどうなのか分かりません。でも、全く動かないものは役には立たないので、これでしばらく使って不都合が出るかどうか確かめてみます。おそらくnvi付属のcursesライブラリが64bit環境では動かないのでしょうね。バグ報告をしてあるので、この記事が出るまでには直っているかもしれません。

完全ディスクレスマシンの構築

●ディスクレスマシンの効用

7月号に書いたSRMコンソールを使ったネットワークブートで、カーネルだけブートする方法は中途半端な気がするという人もいるかもしれません。ですが、FDISKでパーティションを切ったハードディスクやSRMでサポートされていないドライブでも、Linuxならサポートしているというものも多いので、Windows NTとの共存や、古いマシンでIDEドライブを搭載したシステムを作りたい向きには、大変重宝します。フロッピーディスクをブートデバイスとして使う方法だって当然あるのですが、ネットワークからブートしたほうが高速ですし、カーネルの再構築などの対応も簡単です。それにクラスタを構成するような場合、1台1台別々にブートディスクを用意するのも、そのディスクをメンテナンスするのもかなり手間がかかるものです。ネットワークが高速になっている現在、カーネルブートだけでなくファイルシステムもネットワーク上に用意する完全ディスクレスのマシンもそれほど性能的に見劣りしないものになっています。しかもサーバとクライアントが同一のアーキテクチャなら、アプリケーションのインストールもサーバで1回行えば、エクスポートしている先のクライアントでも自動的に利用可能となることで、メンテナンスも容易になります。

●ネットワークブートの設定

というわけで、7月号に引き続き、今月号ではディスクレスのAlphaマシンの構成方法を書いていきます。特に今月は、完全ディスクレスの環境を設定していきたいと思います。

さて、カーネルのネットブートだろうと完全ディスクレスだろうと、サーバがなければネットワークブートはできません。サーバにはbootpかdhcpのパッケージを使うのが通常だと思いますが、Debian/potatoのフリーズ版の初期のbootpパッケージは、残念ながらlongを32bitとして扱うバグのあるソースを使っていて動作しません。パッケージのメンテナに報告しておいたので、この号が出ることには動くようになっているはずです。dhcpのパッケージでサーバを構築すれば問題なく動作しているので、bootpパッケージが修正される前に試してみようと思っている方は、dhcpパッケージをご利用ください。

また、私のところだけの問題なのかもしれませんが、2.2.14のカーネルでは、「Multia」はネットワークブートできませんでした。「EB164」は全く問題なくブートできるし、slinkのパッケージにある2.2.5やMikitaさんのネットワークインストール用のカーネル(2.2.15pre？)では問題なくブートできています。きちんと追いかける必要があるかもしれませんが、2.2.5でも取りあえず困らないので、そのままにしています。

さて、もう一度カーネルのネットワークブートの設定を復習してみましょう。

①TFTPサーバとBOOTPサーバを構築

- bootpパッケージをインストールし/etc/bootptabを設定
- /etc/servicesにbootpcとbootpsの行があることを確認
- /etc/inetd.confでtftpとbootpのコメントを解除
- inetdを再起動

/etc/bootptabにはブートファイルの場所とファイル名を記述しますが、そのディレクトリからtftpでファイルを取得できなくてはいけません。inetd.confに記述するtftpのルートディレクトリからブートファイルを取得する場合には、ブートファイルを置く場所をあえてbootptabに書かずに、ブートファイル名のみを記述するようにしましょう。

②SRMコンソールの設定

・ブートデバイスをewa0に設定

```
>>> set bootdef_device ewa0
```

・プロトコルをBOOTPに設定

```
>>> set ewa_protocol bootp
```

・ブートフラグを適宜設定(以下は私のMultiaの例)

```
>>> set boot_osflag "root=/dev/sda2"
```

・必要ならブートファイルを設定(通常はbootptabで指定)

```
>>> set boot_file "bootpfile"
```

以上の設定を間違わずに行えば、ネットワークブートができるはずです。SRMコンソールでは、ブート時にファイル名やブートフラグをNVRAM経由でカーネルに渡すことができるため、Linuxのブートには大変都合が良くできています。

完全ディスクレスの設定をする場合には、いくつかのポイントがあります。ドキュメントとしては「NFS-Root-Client Mini-Howto」が良くまとまっています。本記事の手順もAlpha固有のところ以外は大体このドキュメントにそったものになっています。

●SRM環境変数の変更

ブートオプションを渡す必要があるので、実行例1のように変更しています。ディレクトリ名やIPアドレスなどは各自の環境に合わせて変更してください。

長い環境変数を渡せない「Jensen」などのシステムでは、abootを使って対話的にブートする必要があるかもしれません。もしくは、カーネルにあらかじめデフォルトオプションを書きこんでおけばいいでしょう。

●ルートファイルシステムをNFSマウント

ルートファイルシステムをNFSでマウント可能にするにはNFSROOTの設定をすればいいので、カーネルのドキュメントのnfsroot.txtに従って、必要な構成を持つカーネルを作成します。このときに、/がマウントされるまでに必要となるデバイスドライバはモジュールにしてはいけないことに注意してください。モジュールを読み込むには、/をマウントしなくてはならないのに、/をマウントするにはモジュールが必要ということに陥ると失敗します。NFSの項目に"CONFIG_ROOT_NFS"が出てこない場合には、ネットワークのオプションに"IP kernel level autocofiguration (CONFIG_IP_ PNP)"を指定していない場合があります。これを設定してさらにbootpとrarpのサポートを追加してください。これらのオプションを設定すると、ブート時にサーバを探してIPアドレスの設定をしてくれるので便利です。しかし一度だけ、1つのネットワークに複数のDHCPサーバがあるときに、違うサーバから勝手にアドレスを取ってくることがありましたので、利用には注意が必要かもしれません。カーネルの構成時の/usr/src/linux/.configファイルに次の行があることを確認してください。

```
CONFIG_NET=y
CONFIG_IP_PNP=y
CONFIG_IP_PNP_BOOTP=y
CONFIG_NFS_FS=y
CONFIG_ROOT_NFS=y
```

また、カーネルコンパイル時に、SRMが認識できるネットワークカードがモジュールではなく、カーネルに組み込まれるようにしてください。「make menuconfig」で対応するエントリが"*"となっていれば大丈夫です。当然ながら前出のオプション群もモジュールではいけません。

●共有可能なディレクトリを整備

次に、複数のマシンをディスクレスとする場合に備え共有可能なディレクトリを整備します。代表的なディレクトリは/usrですね。サーバと同じアーキテクチャのマシンをクライアントにするならば、サーバの/usrをエクスポートして、クライアントがこれをマウントするだけですみます。注意しなくてはならないのは、同じAlpha同士でも、プロセッサモデル

実行例1　SRM環境変数の変更

```
>>> set boot_osflag "root=/dev/nfs nfsroot=Server_ip_address:/remote/client_dir"
```

によってインストールするパッケージが異なるような場合には(コンパックのCPML、CXMLなどが相当します)、この手法がそのまま使えないことがあります。Xサーバなどで行われているように、一度、/etcにシンボリックリンクを飛ばしてから、必要なファイルにシンボリックリンクを張り直す方法がスマートでしょう。

●/etcや/varなどを共有

個別に内容が異なるディレクトリ中で、可能なものを共有します。これには/etcや/varが相当します。全部は共有できないのに、一部は共有したほうが管理の都合上もやりやすいことがありますね。個別に用意する必要があるファイルがあるので、クライアントは、これらを直接相当ディレクトリにマウントすることはできません。

そこで、例えば、クライアントに/clientというディレクトリを作成し、/client/etc、/client/varに共通の/etcと/varをマウントします。その後で、これらのマウントしたディレクトリから共有すべきファイルやディレクトリを個別にシンボリックリンクしていきます。これらの作業は、サーバ上で行うことになりますが、そのときには、サーバには/client/etcや/client/varがないのに、そこへのシンボリックリンクを作成しなくてはならないので注意が必要です。もっとも、サーバにも、/client/etcや/client/varを作っても、当然、構いません。TeX関連ツールが自動生成する、/varにあるフォントファイルなどを共有するには、クライアントがルート権限で書き込みできた方が便利なので、私は、texmfの関連ファイルだけはサーバのものをマウントしています。これらのファイルは書き込み可能にしますが、無制限に大事なファイルをルート権限に開放するのはあまり好ましくありませんから、多少面倒でも必要性のあるものに限定しましょう。また、LinuxのNFSサーバは、ワイルドカードを用いたアクセス制限ができるので、これらもうまく使うことができるでしょう。

●クライアント環境の整備

クライアントは、サーバの/lib、/bin、/usr、/sbinをマウントして使いますから、これらのディレクトリには何もいらないと思いがちです。しかし、スタートアップスクリプトで必要なファイルをマウントする前には、ルートしかマウントしていないので、NFSが稼動し始めるまでに必要なバイナリはあらかじめルートディレクトリの対応する場所に格納しておかなくてはなりません。複数のクライアントを用いるときには、クライアントのファイル同士でハードリンクを張ればいいので、ディスク容量の面ではそれほど負担にはなりません。何が必要となるのかは、後述の/etcのスタートアップファイルの中身によって変わります。私は、表1にあげるようなものを置いてあります。利用するコマンドの必要なライブラリは、lddコマンドで確かめられますが、実際には、動かしながら足りないライブラリを足していったりしました。コマンドも同じように足りないものを足していけばいいのですが、シンボリックリンクになっていたり、スクリプトになっているコマンドは注意が必要です。私も、/bin/shをコピーして安心していたら、リンク先のbashを入れ忘れて、スクリプトが動き出さないという失敗を最初にしています。

●クライアントのスタートアップスクリプト

一番気を使うのが、クライアントのスタートアップスクリプトの整備と作成でしょう。この項では、特に断らない限り、クライアントの/etcに相当するディレクトリのことを書きます。私の構成では/remote/client_dir/etcの下のファイル群です。間違ってサーバの/etcを編集しないようにご注意ください。

さて、NFSでルートファイルシステムは読み込んでいますが、ネットワークやマシンの個別の設定は、このスクリプトの中で実行していくことになります。ランレベルごとのスクリプトのスタートポイントは、/etc/inittabで指定しています。立ち上げ時のスクリプトは、Debianでは/etc/init.d/rcSになります。このスクリプトは、/etc/rcS.dのSから始まる名前のスクリプトを順番に実行していくだけなので、あまり凝ったことはやっていません。そこで、/etc/rcS.dのコマンド群の名前付けが重要になります。個別のスクリプトの構成はディストリビューションに合わせる必要があります。ここでは、Debianを例に説明していきますので、別のディストリビューションを使っている方は、対応する処理を各自で考えてください。

まず、NFSでは、ファイルシステムのチェックはサーバが行っていて、クライアントは行う必要がないため、fsckを止めます。しかし、ルートを書きこみ可能にするremountは必要なので、checkroot.shのスクリプトを書き換えて、fsckを

表1　クライアントマシンに用意しているファイル

ディレクトリ	ファイル
/bin	bash、hostname、mount、rm、sh
/lib	Sld-*、libc-*、libc.*、libdl.*、libdl-*、libncurses.*、libresolv.*、libpopt.*
/sbin	getty、ifconfig、init、route、halt、reboot

リスト2　クライアントマシンの/etc/fstab

```
none /proc proc default 0 0
server_IP:/remote/client_dir / nfs default 1 1
server_IP:/remote/client /client nfs default,ro 1 1
server_IP:/lib /lib nfs default,ro 1 1
server_IP:/bin /bin nfs default,ro 1 1
server_IP:/usr /usr nfs default,ro 1 1
server_IP:/sbin /sbin nfs default,ro 1 1
server_IP:/home /home nfs default 1 1
server_IP:/var/lib /var/lib nfs default 1 1
server_IP:/var/spool/texmf /var/spool/texmf nfs default 1 1
```

行っている部分を削除します。また、checkfs.shは、rcS.dから外します。通常のマシンと違って真っ先にNFSで各種バイナリやライブラリのマウントをしたいので、mountall.shをcheckroot.shのすぐ次に持ってきます。また、mountall.shの中では、mountコマンドはnonfsのオプションがついていて、このままではNFSはマウントされないので、このオプションを削っておきます。mountが動作するためには、/etc/fstabが適切に設定されていなければなりません。私のクライアントの/etc/fstabに書きこむエントリは、リスト2のようになっています。

ネットワークの設定前に、マウントするためホスト名ではなく、IPアドレスを指定してNFSのエントリを作ります。最後の2つのエントリは、TeXのフォントやフォーマットファイルを共用するために必要です。また、networkingのスクリプトで参照される/etc/network/interfacesの中のeth0のオプションを"bootp"に変更します。立ち上げ時に必要なスクリプトの修正はこれだけです。

次はスクリプトの名前ですが、Debian/potatoから変更したものだけを表2にあげておきます。修正は、とにかくマウントするまでは実行できないコマンドを後回しにして、全体として辻褄が合うようにするだけです。ランレベル0と6のhaltとリブート時のスクリプトでは、コマンドがNFSでマウントしたライブラリを呼ぶ状態でアンマウントのリクエストが出るので、/libは使用中でビジーになり、ファイルシステムのアンマウントがうまくいきません。取りあえず実害はないのでそのままにしてあります。気になる人は、/standなどのディレクトリにスタティックリンクのバイナリを置いて、これらのランレベルでは/libを読まないようにしてください。いずれにせよ、NFSではいきなり電源を落としてもファイルシステムを壊すおそれはないので、アンマウントさえ不要かもしれません。

●クライアントの追加

クライアントを追加するには、サーバ上で"cp -a"を実行し、動作中のクライアントのディレクトリをコピーして新たなクライアント用のディレクトリを作ります。ここで、/etcの中身のように、クライアントごとに異なるものは、コピー後に調整が必要になります。せっかく作ったのでコピーするイメージはサーバがダウンしたときに再構成できるように、tarででも固めて、別のディスク(できれば別のマシン)に取っておきましょう。tar+gzipでまとめると2Mbytesもないくらいの大きさです。サーバの方は、インストールしたパッケージ名と/etc/exportsなどの一部のファイルさえバックアップしておけば、代替は容易です。

ここに上げた方法で、現在、EB164とMultiaを完全ディスクレスの形で運用できています。Multiaは静かでコンパクトなので、内蔵ディスクもフロッピーディスクも使わないこういう形態は、よく合っているように思います。

表2　起動スクリプト名の変更

スタンドアローン(Debianオリジナル)	ディスクレスクライアント
S05keymaps-lct.sh	S20keymaps-lct.sh
S30checkfs.sh	K30checkfs.sh
S30setserial	K30setserial
S35mountall.sh	S15mountall.sh
S40hostname.sh	K40hostname.sh

R E S O U R C E

[1] 「Linux 2.0カーネルブック」
R.Card、E.Dumas、F.Mevel 共著／臼田昭司ほか共訳
オーム社／ISBN4-274-07879-5／5300円(税別)

[2] Shizuoka Linux Users Group
活動地域：静岡
http://www.szlug.factory.to/

[3] Linux Seminar Working Group
http://www.linet.gr.jp/lswg/