Linux/Alpha活用講座

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

第18回
Debianの環境設定

インテルやAMDの、1GHzを超えるプロセッサのデモンストレーションに比べておとなしいこのごろのAlphaですが、ようやくUP1000が出荷されるようになって、低価格の高性能マシンが入手できる見込みが出てきました。見積もりだけ取っていますが、安くなったとは言っても何カ月分かの原稿料がすっ飛ぶ価格には違いないので、導入には躊躇があります。見積りを取ったところは、初期出荷の特別割引で約2000ドルとAthlonやPentium IIIとさほど変わらない値段設定になっています。普通に入手する場合には、APIのデベロッパ登録をすることで約3000ドルで購入できるようです。

Alpha Linuxのニュースから

・InformixがAlpha Linux用の商用データベースパッケージを販売します。商用のアプリケーション(特にデータベース)が揃ってくることでビジネス利用への道を開くことになるので、私は単純に喜んでいます。

・APIとRed Hatがクラスタリサーチセンターを開始。当初は32台のデュアルプロセッサマシンでスタートし、研究結果はすべてGPLで公開するということです。株式公開後さまざまな出資を受けて意気揚々のRed Hatが、どんなことをやってくるのか楽しみですね。

・MandrakeSoftもAlphaのディストリビューションをリリース予定。

・CompaqがPower Toolsというサイトを開始しました。

http://www.support.compaq.com/alpha-tools/index.html

ここにはAlpha Linuxで動作する有用なソフトがいくつか集められていますから、Web上を探し回らなくてもよくなります。

・日本のコンパックからAlpha Linux用のFortranとCコンパイラが正式に発表されました。価格はそれぞれ6万8000円、8000円となっています。Fortranのライセンス形態がはっきりしたことで企業ユーザーでも使いやすくなりました。

NetscapeのDebianへのインストール

先月号で最後に駆け込みでCompaqからリリースされたNetscapeのAlpha Linuxバージョンについてお伝えしましたが、Debianのユーザーは多少手作業で設定するところがあるのでその手順を示したいと思います。

1. ニュースのところで紹介したPower Toolsのサイトから NetscapeのRPMファイルをダウンロードします。
2. alienコマンドを使いdebファイルに変換します。

```
$ alien netscape*.rpm
```

3. dpkgコマンドで変換したファイルをインストールします。

```
# dpkg -i netscape*.deb
```

これで、/usr/local/netscapeにtar+gzipされたファイルとRPMのspecファイルが展開されます。

4. tar+gzファイルを展開します。

```
# cd /usr/local/netscape
# tar xzf netscape*.tar.gz -C /
```

5. specファイル中のインストール後の処理に相当する部分を手作業で実行します。

```
# ln /usr/shlib/libc.so /usr/shlib/libc_r.so
# ln -s /usr/shlib/libc.so /usr/shlib/
libpthread.so
# ln -s /usr/shlib/libc.so /usr/shlib/
libpthreads.so
# touch /etc/sia/siainitgood
# chmod 755 /sbin/loader
# chmod 644 /etc/svc.conf
# chmod 644 /etc/sia/*
```

これで無事にNetscapeが動作するはずです。日本語リソースの追加など実用のための作業はしていませんが、取りあえず日本語を含むWebページの閲覧ができるようになりました。

Debian 2.1r4のインストールのお話

私のところのXP1000は、WindowsNTとのデュアルブートの環境を維持するために、Linuxはフロッピーディスクからのブートをしていましたが、NTを使うケースは全くといってよいほどなかったので、ハードディスクのパーティションを切り直して新規インストールしました。新しいディストリビューションもいろいろ出ているので順次試してみたいと思っています。MILOのリリースされていないマシンにおいてLinuxをブートするためには、AlphaBIOSではなくSRMを使う必要があることは以前書きましたが、今回もSRMコンソールを用いてブートします。SRMを使ってハードディスクからブートするにはパーティションはTru64と同じdisklabel形式になっていないといけません。ところが、Debianのインストーラはdisklabel形式を作るようになっていません。おそらくこの点がインストールの一番のポイントでしょう。また、私の利用環境ではNISのパッケージがきちんと動作しなかったので、コンパイルし直す必要がありました。インストールの手順は次のようになります。BIOSの設定など細かな点は、本誌1999年11月号を合わせて参照してください。

①Debianのgenericというブート用フロッピーディスク用意します。必要なディスクは3枚で、それぞれresc1440.bin、drv1440.bin、root1440.binという名前のファイルから作ります。

②resc1440.binでブートします。途中でRAMDISKを要求されたらroot1440.binに入れ替えます。ブートはSRMのプロンプトから次のように行います。

```
>>>boot -fi linux -fl "root=/dev/fd0
load_ramdisk=1" dva0
```

③Debianのインストーラが起動しますが、1回目はインストーラではなくシェルを使ってパーティションを切ります。ALT-F2を押して画面を切り替えてください(ALTキーを押した状態でF2キーを1回押します)。画面切り替え後Enterキーを入力すると、「#」というプロンプトが出てシェルが起動したことが分かります。

④fdiskを起動します。fdiskはそのままではDOS形式のパーティション作成のモードになりますから、bコマンドを入力することでBSD disklabel形式にモードを切り替えます。パーティションの作成において注意する点は次のようなところです。

- fdiskではアルファベットでパーティションの番号を指定します。カーネル自身は番号そのものを使うので注意してください(fdiskでaパーティションを作るとカーネルはたとえばSCSIディスクなら/dev/sda1として扱います)。一般のUNIXの使い方として次のようにすることが多いでしょう。

  a: ルートディレクトリ
  b: スワップファイル
  c: 全パーティション
  d〜h: ユーザー等各種パーティション用

- c以外のパーティションの大きさを最大でも4GB以内に抑えてください。mke2fsでのファイルシステムの作成は成功したように見えるのですが、fsckが通りません。
- ブートローダー用に、若干のシリンダをあらかじめファイルシステムに使うパーティションへの割り当てから外します。たとえばaパーティションに全部のパーティションを入れるならばaを第1シリンダからではなく第2シリンダから開始するなどとします。全ドライブを示すcパーティションにはブートローダーのシリンダを入れても構いません。
- パーティションを作成したら、忘れずにエントリのファイルタイプを指定しておきましょう。
- パーティションを設定したら忘れずにwコマンドでdisklabelに書き込んでおきましょう。

⑤disklabelを反映させるために一度リブートします。インストーラの状況によってはリブートは不要かもしれませんが、何回かやってみたところリブートしないと反映されないケースがあったのでリブートをしたほうが安心です。

⑥インストーラの手順に従ってインストールをします。ネットワーク経由でインストールする場合、ベースファイルだけは何とか持ってくる方法を作らなければなりません。私はNFSサーバにbase2_1.tgzを置いてNFS マウントして使う方法を取りましたが、必要に応じて分割されたファイルを使いフロッピーディスクベースで利用するなど工夫してください。

⑦ベースシステムを展開したらリブートしますが、このときにはまだハードディスクにブートローダーをインストールしていないので、フロッピーディスクからのブートになります。fdiskのパーティションの名前とブート時にカーネルが認識するパーティションは異なっていることにもう一度注意してください。resc1440.binのFDを入れてSRMのプロンプトから次のように入力します(ここではaパーティションにルートファイルを作成したと仮定します。そうするとカーネルからは/dev/sda1がルートになっているように見えます)。

```
>>> boot -fi linux -fl "root=/dev/sda1" dva0
```

⑧Debianには複数の構成の候補があって、インストールの最初でどれかを選択するようになっています。私は「Standard package」を選択して必要なファイルをその都度追加するようにしました。こうすると、最初は150MB程度のダウンロードですむからです。

⑨次のリブートに備えてハードディスクにブートローダーをインストールします。

```
# cd /boot
# swriteboot -f 3 /dev/sda bootlx
```

この"-f 3"の引数は3番目のパーティションはブートローダーと重なった場所になることを示します。これ以外のパーティションを第1シリンダから確保したときには、その番号を合わせて指定しておいてください。

⑩日本語や日常使うツールを使うためにいくつかのパッケージを追加します。私が追加したパッケージは次の通りです。コンパック提供以外のパッケージはapt-getコマンドを使って依存関係にあるパッケージを含めてインストールします。

X関係：
fvwm95、tgif-ja、xcontrib、xdm、xfonts-75dpi、xfonts-base、xfonts-cjk、xfonts-scalable、xfs、xlockmore、xserver-mono、xlib6g-dev

その他のユーティリティ：
fetchmail、imagemagick、xntp3、alien、lha、rpm、unzip、wget、nis

日本語環境：
a2ps-ja、canna、jless、kinput2-canna、kterm、mnews、nvi-m17n-canna

TeX環境：
dvipsk-ja、gs-aladdin-vflib、platex、tetex-extra、tetex-nonfree、xdvik-ja

数値計算環境：
g77、lapack、lapack-dev、lapack-doc、octave

コンパック提供：
ccc、libots、cpml-ev6、cxml-ev6、cfalrtl-rh52、cfal、ladebug、netscape

コンパック提供のFortranパッケージはDebianのサポートがありますが、ccc(Compaq C Compiler)や、NetscapeはRPMファイルしかないのでalienコマンドで展開してからインストールします。cccはβ版と同じくalienは失敗するので、展開してdebian/rulesファイルの編集が必要です。cccのインストールの詳細は本誌1999年12月号を参考にしてください。Debian 2.1r4のパッケージにはXP1000のビデオカードをサポートするXサーバが入っていないので取りあえずXの環境はX端末から使っています。potatoではxserver-3dlabsを導入することでコンソールで普通にXが使えます。

⑪カーネルのバージョンがslinkは2.0.36になっていますが、2.2系列のカーネルでないと不便なことが多いのでカーネルをダウンロードしてカーネルのインストールをします。また、この時についでにresource.hの中のスタックサイズの調整をしてしまいましょう。以前この問題をメーリングリストで指摘したのですが、Alphaのソースは未だに変更されていません。いつのまにかx86やppcは指摘した通りに変わっているので、もう一度リクエストしないといけませんね。resource.hの28行目をエディタで次のように変更します。

```
  {_STK_LIM, LONG_MAX}, /* RLIMIT_STACK */ \
```

この構造体のメンバーはデフォルトのスタックサイズと最大のスタックサイズを規定しています。配布ファイルはどちらも_STK_LIMになっていますが、最大はLONG_MAXにしておいてください。カーネルのパッケージを取ってきてコンパイルする手順は次のようになります。

```
# apt-get install kernel-source-2.2.5
# cd /usr/src
# tar xzf kernel-source-2.2.5.tar.gz
# ln -s kernel-source-2.2.5 linux
# cd linux
# vi include/asm-alpha/resource.h
# make menuconfig
# make dep
# make boot
# make modules
# make modules_install
# mv arch/alpha/boot/vmlinux.gz /
```

⑫一度シャットダウンしてSRMコンソールの環境変数を設定し、ブートの確認をします。

```
>>> set boot_flags "root=/dev/sda1"
>>> set boot_file vmlinux.gz
>>> set boot_device dka0
>>> set auto_action boot
>>> boot
```

⑬NISのパッケージはなぜか私のところではきちんと動かなかったのでソースファイルを取ってきて作りなおしました。rootでログインして次のコマンドでソースを入手します。

画面1　netlibのトップページ

```
# apt-get source nis
```

パッケージの作成とインストールを行います。

```
# cd nis-3.5
# debian/rules binary
# dpkg -i ../nis*.deb
```

以上の手順で普段使う環境は整ってきています。まだ使い込んでいないので、この後必要なパッケージがいくつか出てくると思いますが、必要になるたびにインストールしていきたいと思います。Debianが提供しているパッケージの中でtgif-jaだけは私は自分でpotato環境でコンパイルしたバイナリに置き換えています。おそらくコンパイル環境の問題だと思いますが、Debian 2.1r4のAlpha版のtgifは図の矢印が画面上表示されないバグがあることと、tgifの日本語コードがEUCになっていて一部のPSプリンタと相性が良くないためです。

A Compaq FortranとCXMLを使ったベンチマーク

正式に日本でもコンパイラがリリースされたので同時にリリースされている数値計算のライブラリの性能も含めてベンチマークを取ってみましょう。ベンチマークの課題はLINPACKベンチマークで1000元の連立一次方程式を解くものです。

```
http://phase.etl.go.jp/netlib/benchmark/1000d
```

Netlibにある1000元のベンチマークプログラムを使って性能を確かめてみます。このベンチマークを動作させるためにはsecondという時間計測関数を作らなくてはいけませんが、その中身は本誌2000年1月号の記事を参照してください。CXMLはLAPACK互換なのでLAPACKを使うようにベンチマークプログラムを変更します。元々はLU分解法で方程式を解いているので分解と求解のサブルーチンを別々に呼び出していますが、今回は1つにまとめてLAPACKのDGESVの呼び出しに置き換えてしまいましょう。1000dのファイルの次の部分がベンチマークの対象になっています。

```
t1 = second()
call dgefa(a,lda,n,ipvt,info)
time(1) = second() - t1
call dgesl(a,lda,n,ipvt,b,0)
time(2) = second() - t1
```

この部分を次のように書き換えます。

```
time(1) = 0
t1 = second()
call dgesv(n,1,a,lda,ipvt,b,n,info)
time(2) = second() - t1
```

LINPACKとCXMLとLAPACKでのベンチマーク性能(MFLOPS値)を表1に挙げます。LAPACKのパッケージは残念ながらg77の内部関数を使っているようで、CompaqのFortranではいくつかの関数が未定義というエラーが出てしまってg77でしか評価できませんでしたが、ご覧のようにこの問題ではXP1000ではLAPACKのほぼ4倍の性能を出しています。21164の世代ではさらに性能差は開いていて10倍以上の性能になります。ライブラリをうまく使うことの重要性が分かると思います。LAPACK自体をコンパックのFortranでコンパイルしなおして評価したらどうなるか興味深いところですね。密行列の計算ルーティンはCXMLではかなり細かなチューニングをしているので21164AのVT 5-600の性能は500MHzのクロックのXP1000よりも高くなっています。

それではこの結果をもう少し詳しく解析してみましょう。こういったプログラムの解析に力を発揮するプロファイラを使います。まずは、ターゲットのプログラムを"-pg"オプションをつけてコンパイルします。

```
$ fort -pg -O6 -o 1k.p 1000d.f second.f
```

次にこのプログラムを実行します。

```
$ ./1k.p
```

すると、gmon.outというプロファイルのねたファイルができますので、これをgprofで表示します。

```
$ gprof 1k.p gmon.out
```

どうですか？　関数ごとの実行時間が分かりやすい形で示されているはずです。例えば、LINPACKをfortでコンパイルしたときに関数の実行時間の長い順に上位から並べると表2のようになります。LINPACKベンチマークがDAXPYベンチマークだといわれる理由が良く分かりますね。次にfortでCXMLを使ってコンパイルしてみます。シェアードライブラリを使うとプロファイルルーチンを挿入できず時間の計測ができないので、スタティックライブラリをリンクします。

```
$ fort -pg -O6 la1k.f second.f /usr/lib/
libcxml.a -o la.p
$ ./la.p
$ gprof la.p gmon.out
```

先ほどと同じように時間の上位から表示してみたのが表3です。

これ以外にも呼び出されているルーチンはありますが、ここに上げたもので94%以上の時間をしめているので、大体の特性は把握できると思います。どちらにも出てくるmatgenはベンチマークをする前に行列を生成するためのサブルーチンなのでここではベンチマーク数値に関係はしません。CXMLをリンクしてコンパイルすると一番重要な関数が行列行列積のサブルーチンであるdgemmであることが分かります。CXMLのdgemmは日本からコントリビュートしたものが採用されていて高速化に寄与しています。

おわりに

Kondara MNU/Linuxのリテールパッケージを購入しました。このパッケージにはx86版とともにAlpha版も入っているので手軽に入手できるAlpha用のパッケージとして期待できます。x86版はインストールしてみましたが、時期的に研究室のAlphaの使用頻度が高く、Alpha版のインストールテストはできていません。近いうちにお伝えできると思います。

表1

マシン	Linpack (fort)	Linpack (g77)	lapack (g77)	CXML (fort)	CXML (g77)
XP1000-500	99.23	83.61	148.2	579.8	531.6
EB164-300	31.96	20.47	30.56	377.7	307.3
VT 5-600	58.84	37.51	56.83	669.3	563.6

(単位はMFLOPS)

表2

項番	関数名	時間割合	実時間(EB164)
1	daxpy	96.00%	21.15秒
2	dgefaq	1.85%	0.41秒
3	matgen	1.68%	0.37秒
4	idamax	0.23%	0.05秒
5	dmxpy	0.21%	0.05秒

表3

項番	関数名	時間割合	実時間(EB164)
1	dgemm_nn	59.14%	1.51秒
2	matgen	13.85%	0.35秒
3	dlaswp	9.30%	0.24秒
4	dgemv_n	8.49%	0.22秒
5	dtrsv	3.37%	0.09秒