Linux Japan 1999年７月号(?)原稿

修正した原稿を送ります。
修正に併せて最後に二つ目のコラムを追加しました。

Linux/Alpha活用研究

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

前号でWORDの添付ファイルの話を書いたのですが、同じ号のよしだ
ともこさんの記事に解決方法が出ているではありませんか :-) ま
だ試していませんが、期待が持てます。Interixの方も少しずつ環
境を整えてきましたが、POSIX互換といってもなかなか自前で環境
を立ち上げるのは難しいものです。Tgifのコンパイルは何個所かのパッチ
でできるようになったのですが、日本語のテキストを入れるまでに
はなっていません。Ｘのセレクションからペーストすれば楽勝と思
っていたらTgifの方にその機能がないのでした ;_;
kinput2とCannaを入れれば使えそうですが、漢字変換を別に持つのも
芸がないので(さほど緊急性も高くないこともあって)そのままに
なっています。(授業のOHPは絵の中は英語で書くことにしました)

さて、COMPAQから次なる贈り物がLinuxコミュニティに届きま
した。それはポータブル算術ライブラリです。従来NetBSD/OpenBSD
やLinuxのglibc-2.0までの算術ライブラリはＳＵＮが公開したもの
を使っていましたがこれは設計が古く性能的に満足いくものではあ
りませんでした。算術ライブラリを使わなくても関数演算の多くを
実行できるi386系のＦＰＵに最新のＲＩＳＣプロセッサがなかなか
かなわない理由もここにありました。glibc-2.1では私の提供
したパッチも入り基本演算の性能はそこそこに上がっていますが、
それでもまだ手を加えるべき点も多いのです。COMPAQのライブ
ラリは完成度の高いものになっているようです。これで後はコンパイ
ラさえ出てくればというところですね。 :-)
ちなみにこのライブラリのダウンロード件数ですが日本は米国に続いて
第二位だそうです。関心を
持たれている方の多さがわかりますね。ライブラリは次のＵＲＬを
たどって入手してください。

http://www.unix.digital.com/linux/cpml.htm

COMPAQがコンパイラを出すという噂はあるようですが、
この雑誌が出るころにはCOMPAQからLinux/Alpha用コンパイラ
のリリースなんてニュースが出てくることを期待しましょう。

一台では足りない性能を補うのに複数のプロセッサの並列処理を
行なうことは数値計算の分野では常識になっていますが、DS20の
ようなマルチプロセッサのマザーボードでなくても単にイーサーネット
で接続しただけのマシンでもそこそこ動くというのは'99年1月号でAvalonを紹介
したときに書きました。しかし、いくらATXのスリムなマザーボードだといっても
これを数多く並べるのは場所を取ります。この点に目を付けて簡単に並列システムを
作るためのケースをイギリスの会社が販売しています。

http://www.siamese.co.uk/

の8-packというケースを見てください。最近のプロセッサの発熱は
尋常ではないのですが、このシステムはAlphaをターゲットの一つと
して宣伝しているので放熱の設計もされているものと思います。
ケースだけで 800.00は高いと思う人もいるかもしれませんが、
システムがコンパクトにまとまるので優れた方法と言えるでしょう。

さて、日本のLinux/Alphaのコミュニティを長い間支えていたメイ
リングリスト Linux-Alpha-JP が管理サイトのドメイン変更などの
事情で廃止されることになりました。長い間管理人を勤めてくださ
った公文さんに感謝します。どうもご苦労さまでした。

＊＊＊＊＊＊ コラム ＊＊＊＊＊＊

年次報告書を読もう

さて、COMPAQ社がどこまでアルファに本気なのか疑う人は多い
し社外からは（おそらく社内にいても）本当のところは分からない
ものでしょう。COMPAQの年次報告書が３月に発行されました。
私はハードコピーを一部持っていますが、この本文は

http://www.compaq.com/corporate/ir/index.html

で見ることができます。この中から少し関連事項を抜き出して将来
動向を推測してみましょう。

Digitalの開発中の技術に対する投資額は３２億ドルとなっていま
す。これは９８年度の売り上げの１０．２％に相当します。一方、
粗利益率は約４％下がって、人件費負担は４％上がっていま
す。開発中技術の購入費用は今年度で償却されるとすると人件費と
粗利益率を何とかしなくてはなりません。粗利益率の
中身を見てみると製品の利益率は６％も減っていて２１．９％に
なっているのに対してサービス部門は３１．６％の利益率と昨年
実績より良くなっています。さらに２％ほどしかなかったサービス
部門の売り上げが９８年度は１５％ほどに上昇していることからDigital
の買収のうまみがサービス部門に端的に表れているのは間違いない
でしょう。製品部門はＰＣの低価格化を考えると収益性の向上は簡
単ではないでしょうから、COMPAQがさらなる生長を求めるには
サービスに注力していくこと必要となることが良く分かります。
しかしCOMPAQはフルサービスのエンタープライズコンピューティング
企業として開発投資を続けると言っているのでトップエンドのマシンの
開発は欠かせません。

COMPAQは投資金額に対してコスト割引モデルを使って投資の採
否を決定するとしています。COMPAQ自身の成長率を年率１２か
ら１４％と見込んでいますが、投資に対するリターンはUnix/OpenVMS,
NTに対しては（要するにAlphaの技術でしょう）年率２２％を要求
し、記憶装置に対しては年率４０％を要求しています。選択の結果
は書かれていませんが、投資額は１９９９年から２００２年にピー
クを迎えそれぞれ５億ドル以上の投資をすることになっています。
記憶装置の投資にどれだけまわすのかは分かりませんが、
製造設備を手放したことを考えればこの金額で十分21364の開発費はまかな
えると思います。２００３年以降は大幅に投資金額が減っていきま
すがこれは技術の世代交代によるものです。

面白いことに第四四半期に買収技術を再評価したところ３２億ドル
で購入した開発中の技術は５７億ドルの価値があるとされているの
です。この中でUnix/OpenVMSは１６億ドル、NTは８億ドル、記憶装
置は２７億ドルとなっていて記憶装置に対する評価額がもっとも大
きくなっています。この評価額は前出の割引率を引き当てた後の金
額となっています。購入金額に対して割引後の評価額をCOMPAQ
の成長目標と比較すると全体金額では２００３年程度までの開発投
資を正当化するだけのものになっていると思います。残念ながら買
収時の個別の評価額が出ていないので個々の案件についてどう評価
されているのかは不明ですがマクロにはDigitalの開発中技術はコ
ンパックの基準に十分適合しているといえるので、開発を中断する
理由にはなりそうもありません。

個人的には記憶装置のような差別化が難しく利益率の低い市場への
投資は将来の負担になりそうなので評価額が大きくても今のうちに
売却してしまったほうが得な気がします。が、これは私のバイアス
のかかった見方でしょうね :-)

直接の答えは見つかりませんでしたが、年次報告書を読ん
でいろいろと状況を知っておくことも何かの役に立つと思います。
皆さんも気になる企業があったらぜひその企業のＷＥＢのインベス
ターリレーション(投資家情報)を訪れてみてください。
アメリカの企業は大抵年次報告書をホームページに載せています。

＊＊＊＊＊ コラム終わり ＊＊＊＊＊

Debian 2.1 のインストール

さて、前回 lprobe のために Debian 2.1 のインストールの試行中という話を書きましたが、Debian 2.1 はその後無事リリースされました。ぜひ Alpha の有力パッケージとして育てていきたいですね。
Debian を入れると何が嬉しいかというとアップグレードが簡単になるはずであるということに尽きます。
パッケージの管理なら RedHat でもある程度実現されていますが、システムのアップグレードに対しては Debian の方がずっと容易に可能です。提供されるパッケージの数もすごく多く一般的な用途には困らない程度のパッケージが入っていると思います。

また、Debian-jp のプロジェクトによって日本語のパッケージも次々 Debian 本家の配布にマージされてきており、ＢＳＤの世界の FreeBSD のような様相を呈しています。

Debian 本家の配布にすでに日本語の locale もあるのですが、５月号の特集にも書いてあるようにこれは使わず wmbcs を入れます。新しく入手した EB164 にこのパッケージを入れることにしました。ベースのインストールは５月号に書いたとおりですが、正式リリースに伴いディレクトリが若干移動していますので、ご注意ください。今回は Scientific workstation のセットを入れます。これはかなり大きなセットとなるのでディスクの容量に余裕がない人は必要なものだけに限定したほうが良いと思います。ftp でのダウンロードも大変で私の所では一晩かかっても終りませんでした。
slink-jp が正式リリースされていない状況なのでまだ日本語については十分利用できるとはいいにくいのですが、私のメインマシンとして使用しています。
ただし、XFree86 のサーバーによっては安定性に問題のあるものもあるようで、私も Matrox Millenium を使うと xon で動作させている Netscape を操作中に固まることが多く、S3 のボードに変更しました。一ヶ月ほど使っていますが、極めて安定しています。

執筆時点では配布されているパッケージの中で kterm 上で唯一まともに日本語が使えるエディタは jed ですが、私は普段 vi タイプのエディタを使っているので jed を常用とするのは若干困難です。この原因を探っている最中ですが、ldd コマンドで確認すると日本語を使えるエディタで jed だけは ncurses をダイナミックリンクしていないようなので ncurses のバージョンか terminfo の問題なのかもしれません。
自分で jelvis-1.8 をコンパイルしたところこちらは何の問題もなく動作したので、ncurses の使い方の癖があるのかもしれません。本当ならせっかくコンパイルしたのだから Debian のパッケージにしてプロジェクトに貢献すべきですが、パッケージの作り方を良く理解していないのでローカルに使っています。

通常使うものとしては platex や tgif や Canna がありますが、これらは問題無く動作しているので普段の環境は完全にこちらに変更してしまいました。
latex のパッケージの内良く使うマクロなどは non-free の所にあるので忘れずにインストールしておくと Seminar などのパッケージも使えるようになります。

x86 とユーザーの絶対数が違うのでポーティングのスピードに差がでてしまうのですが、皆なで少しずつでもポーティングしていけば必要なソフトはすぐに品揃えができてくると思います。メイリングリストはなくなってもユーザーでぜひ盛り上げていきたいと思います。

コンパックポータブル算術ライブラリ(CPML)の使い方

COMPAQ がリリースしてくれた算術ライブラリを早速使ってみましょう。
インストールの方法は簡単です。
前出の WEB ページからアーカイブ cpml.0.1.tar.gz を取ってきて展開するだけです。
この中には次のようなファイルが入っています。

```
drwxrwxr-x hanek/hanek       0 1999-03-03 03:30 cpml.0.1
-r--r--r-- hanek/hanek    8066 1999-03-03 04:39 cpml.0.1/README
-rwxrwxr-x hanek/hanek 1485232 1999-03-03 03:30 cpml.0.1/libcpml.a
-rw-r--r-- hanek/hanek    2604 1999-03-03 03:30 cpml.0.1/cpml.h
```

COMPAQ では精度を落として性能を上げたエントリも用意してあり
そちらを使うときには cpml.h をインクルードして使いますが、普通は
もともと備わっている math.h を使えばよいでしょう。
Debian ではユーザーが入れるパッケージは/usr/local 以下に入れることを
推奨しているので、私はこのアーカイブを /usr/local/lib に展開しました。

折角ですからベンチマークを簡単にやってみましょう。
マシンは EB164(21164-300MHz)です。

コンパイル時にリンクするライブラリを切り替えることにします。
まず、通常のコンパイルでは

```
cc -o cptest cptest.c -lm
```

として libm をリンクします。

一方、cpml をリンクする場合には

```
cc -o cptest2 cptest.c -L/usr/local/lib/cpml.0.1 -lcpml
```

とします。Makefile などに書いておけば手間はあまりかかりません。

いつくかのベンチマーク結果(実行時間)を次に示します。

	libm	libcpml
SQRT	4.239082e-06	3.600121e-07
ATAN	1.350641e-06	4.601479e-07
COS	1.090765e-06	3.898144e-07

これを見て分るように libcpml の関数はもとの libm より一桁高い性能になっています。
COMPAQ によると高速と言われていた libffm より高精度でほとんどのルーチンは
libffm より高速だということです。

math.h の換りに cpml.h を使うとさらに高速になるのですが、精度が落るので
通常はこのままの方がよいと思います。
私の環境では ATAN ではコンパイル時に最適化のオプションを付けると
リンクに失敗しました。
これを調べてみると<math.h>の中に次のような記述があって最適化のオプション
を付けたときだけ <__math.h>がロードされるようになっています。

```
/* Get machine-dependent inline versions (if there are any).  */
#if (!defined __NO_MATH_INLINES && defined __OPTIMIZE__) \
    || defined __LIBC_M81_MATH_INLINES
#include <__math.h>
#endif
```

この中で

```
extern __inline double
atan (double __x)
{
  extern double __atan2 (double, double);
  return __atan2 (__x, 1.0);
}
```

となっていて atan は__atan2 のコールに置き換えられています。
__atan2 は cpml には入っていないので最適化すると atan が失敗することに
なってしまいます。(誤解のないように念を押しておきますが、atan2 は
当然入っています。__atan2 という特殊なエントリがないだけです。)
cpml を呼びだすときには__atan2 への置き換えを防ぐには /usr/include/__math.h を
直接書換えるしかなさそうです。しかし、この置き換えは意味がなさそうなのに
どうして入っているのでしょうかねぇ。libm を使う人でもこの定義を潰しても
きちんと動くのでこういった役に立たなく有害な定義はさっさと書換えましょう。

CPML のベンチマークに用いたプログラムは次の通りです。
このプログラムでは時間測定に gettimeofday(2) を用いています。
以前の記事において説明したclocks()やRPCC命令はLinuxではマルチプロセッサで
タスクスイッチが発生したケースにおいて問題を発生する場合がありますので、
264DPのユーザーは若干時間の精度は荒くてもgettimeofday(2)を
使ったほうが安心です。本来はソフトウェアから観測できるリソースは
一貫性を保たなければならないので、この問題点はＯＳのバグ
と言ってもよいと思います。私自身は複数プロセッサのマシンを持っていないので
本当に問題無いかどうか確認はできていませんが、gettimeofday(2)を用いることで
容易に回避できるはずです。ただし、この問題はCOMPAQやCygnusの技術者を交えて
議論したのでそのうちカーネルパッチが出てclocks()やRPCC命令が安全に使える
ようになるかもしれません。

```
---------------------- benchmark program ------------------------------
#include <math.h>
#include <sys/time.h>
#include <unistd.h>

#define FUNC sqrt
#define REPEAT 10

void main() {
 double a,b,stime,etime;
 int i,j;
 struct timeval start, end;

 a=2.0;
 gettimeofday(&start,NULL);
 for(i=0;i<REPEAT;i++) {
         b=FUNC(a);
         b=FUNC(b);
         b=FUNC(b);
         b=FUNC(b);
         b=FUNC(b);
         b=FUNC(b);
         b=FUNC(b);
         b=FUNC(b);
         b=FUNC(b);
         b=FUNC(b);
         b=FUNC(b);
         }
 gettimeofday(&end,NULL);
 etime = (double)end.tv_sec + (double)end.tv_usec*1e-6;
 stime = (double)start.tv_sec + (double)start.tv_usec*1e-6;
 printf("Exec time of FUNC is %e seconds\n", (etime-stime)/(REPEAT*10.0));
 exit(0);
}

-----------------------------------------------------------------------------
```

Iprobeのインストール

さて、先月号でIprobeの概要をお伝えしましたが、これを実際に使うために
インストールしてみましょう。
インストールの方法はいくつかありますが、簡単な方法としてはコンパイル済みの
バイナリを用いることができます。コンパイル済みのバイナリはRPM形式で提供
されるので、Redhatではそのまま使えますが、Debianの場合にはalienコマンド
でdpkg形式に変換してから使いましょう。alienをインストールしていない方は
dselectで追加インストールしておいてください。

```
alian iprobe_suite-4.0-1.alpha.rpm
```

とすると

```
iprobe-suite_4.0-2_alpha.deb
```

というパッケージができます。そこで

```
dpkg --install iprobe-suite_4.0-2_alpha.deb
```

とすればインストールができます。
ツールだけインストールしても動かないので、カーネルモジュールを
忘れずにインストールしましょう。カーネルモジュールは2.2.0pre#4 向け
にコンパイルされているので insmod で組込めないこともあるでしょう。
その場合には自分でコンパイルしなくてはいけません。

自分でコンパイルする場合には bfd ライブラリが必要になりますので、
Debian の場合には binutils-dev をインストールしてください。
私は次のようにしました。

1. kernel を 2.2.1 に上げる。
2. /usr/src/kernel-2.2.1/include/{asm,linux} --> /usr/include
   /usr/src/kernel-2.2.1 --> /usr/src/linux
のシンボリックリンクを張る。(元のディレクトリは違う名前に付け替えておく)
3. binutils-dev をインストール
4. 次のように make

```
make driver lib iprobe rep ipreduce OSTYPE=Linux
make install OSTYPE=Linux
```

その結果は /usr/local/bin と/usr/local/lib に入りますが、
カーネルモジュールは iprobe を展開したディレクトリの下の
drivers/obj/linux_iprobe_driver.o
にあるので、insmod できるように自分で移動しなくてはなりません。

さて、実は Iprobe のパッケージは現在は
EV4 の世代か SRM のコンソールを持つマシンでしか動きません。
これは性能カウンタ用の PAL コード命令が EV5 以降の MILO には
入っていないからで、実行を試みても命令例外で実行できません。

ちなみに、この手の問題を検証するには syslog が役に立ちます。
今回は MILO への wrperfmon 命令のポーティングが終わっていると思っていたので
EB164 で最初に実行して動かないというはめになりました。
syslog には次のように表示されています。

```
Mar 31 18:03:20 et7wnt kernel: iprobe(394): Instruction fault 4
Mar 31 18:03:20 et7wnt kernel: pc = [<fffffe0000027894>]  ra = [<fffffe00000268f8>]  ps = 0000
Mar 31 18:03:20 et7wnt kernel: r0 = 0000000000000036  r1 = 0000000000000000  r2 = 000000000000f000
Mar 31 18:03:20 et7wnt kernel: r3 = 0000000000008000  r4 = 0000000000000001  r5 = 0000000000000005
Mar 31 18:03:20 et7wnt kernel: r6 = 0000020000511260  r7 = 0000000000000680  r8 = fffffc00040b4000
Mar 31 18:03:20 et7wnt kernel: r16= 0000000000000003  r17= 0000000000000000  r18 = 000000000000000
0
Mar 31 18:03:20 et7wnt kernel: r19= 000000011ffff3f8  r20= 0000000000000007  r21 = 000000000000000
3
Mar 31 18:03:20 et7wnt kernel: r22= fffffc00040b8000  r23= 00000200002a8e68  r24 = 000000000000000
8
Mar 31 18:03:20 et7wnt kernel: r25= 0000000000000040  r27= fffffe0000027880  r28 = 000000000000000
1
Mar 31 18:03:20 et7wnt kernel: gp = fffffe0000032300  sp = fffffc00040b7ed8
Mar 31 18:03:20 et7wnt kernel: Code: 46310410  46520411  00000039 <b4130000> 47e23400  24000067  a
75e0000  43c2141e  6bfa8001
Mar 31 18:03:20 et7wnt kernel: Trace: [sys_ioctl+576/640] [strace+76/92]
```

ここで大事なのは iprobe で命令例外が発生したという情報と例外アドレスが
kernel: Code:の項に書かれているのですが、◇で囲まれた命令ではなくその前の
命令が例外を発生しているということです。 00000039 の命令はマニュアルで
調べると PAL の wrperfmon 命令ということが分ります。そこで、この命令を実装して
いないための命令例外が発生したと理解できるのです。

syslog には例外アドレスのところにわざわざ印が付いているのですが、
例外で報告するアドレスと実際に例外が発生するアドレスはアーキテクチャによって
は異なります。
ついでながら、命令例外や特権例外が発生したところで割り込み処理ルーチンで

適切に処理を行なうと仮想計算機が構築できます。

MILOをサポートしているJay A Estabrooさんによると、すでに
MILOへのwrperfmon命令の組込みは完了していてリリース待ちだと
いうことですから雑誌が出るころまでにはなんとかEV5でも使えるようになって
いると思います。
SRMコンソールから(MILOを使わずに)ブートしている方は問題ないはずです。
EV4の世代(21064,21066,21064A等)のMILOではPALコードにすでに
必要な命令が入っているので上述の通り使えます。
私は普段使っているAlphaStation 200 4/100のkernelを2.2.1にする
つもりでいたのですが、2.2.1の要求するリソースをまとめて書き出して
いるうちにこれは全部取り換えた方が早そうだと思いslinkをインストール
してしまうことにしました。前回のEB164に続いて２回目となるので
インストール自身はスムースにできました。(もっとも回線が遅くて時間が
かかる点は改善しようがありませんが..)
標準のカーネルはIprobeにはバージョンが古過ぎるのでkernel-2.2.1の
パッケージも入れてカーネルをアップグレードしましょう。
次号ではこのAlphaStationを使って
実際にIprobeを使って性能の測定をしていきたいと思います。

すこしだけ21264の性能の報告

私のホームページに置いてあるDGEMMベンチマークを何人の方かで実行いただき
最新の21264を含む結果をもらっているので、簡単に紹介します。

http://shimizu-lab.et.u-tokai.ac.jp/pgm/gemm/index.html

このベンチマークは1000x1000の行列乗算のものですが、高速化の技法としてRISCプロセッサ
で有効とされる小行列のコピーをあえて行なわず、TLB性能が悪いマシンでは
性能が出ないような構成を取っています。これはキャッシュを意識するのは
今の世代のプログラムにとって常識であってもTLB(COMPAQ用語ではDTB)の
構成はプログラムに意識させてはいけないと思うからです。
現状のRISCプロセッサのTLBの構成は大型コンピュータの常識では
考えられないくらい貧弱なのですが、OSやプロセッサが少しでも
頑張るとこのデメリットは容易に克服できます。
そのためLinux/AlphaにもぜひLarge Pageを導入したいと私は考えていますが、
まだ基礎調査の段階でどこまでできるかはよく分りません。このプロジェクトが
完成すればわざわざ余計なコピーをした方が性能が良くなるなんてことには
ならないだろうと思っています。

と、前置きはこれくらいにしてベンチマーク結果を見てみましょう。

プロセッサ, OS, コンパイラ, (オプション), 性能

21264-500(DS20), OSF1 4.0, DEC C V5.8-009,(-O4 -fast -tune ev6 -Dev6 -DMAIN), 612.269 MFLOPS
21164A-600(LX164), Linux, egcs-2.90.29,(-O2),359.379MFLOPS
21164A-600(?), OSF1, DEC C,(-O2),317.473MFLOPS
21164-300(EB164), Linux, egcs-2.91.60,(-O2 -DKMNLOOP),185.681MFLOPS
21164-300(EB164), Linux, egcs-2.91.60,(-O2),182.182MFLOPS
21064A-300(AlphaPC64), Linux, egcs-2.90.21, (-O2 -Dev4), 80MFLOPS
21064A-100(AS200 4/100), Linux, egcs-2.90.21, (-O2 -Dev4), 32MFLOPS
PA8000-180(HP D series), HPUX, c89, (+O4 -DKMNLOOP -DAil_max=72 -DBlj_max=8), 242MFLOPS
R10000-200(NEC UP4800), UNIX_SV, cc, (-Kopt=2), 32MFLOPS
Celeron-300A(450MHz), slakware-3.6, gcc (-O2), 136.986 MFLOPS
PentiumII-450, RedHat-5.1, egcs-1.1.1 (-O2), 148.038 MFLOPS
PentiumII-400, WindowsNT, VisualC++6.0 (-O2 -G6), 114.593 MFLOPS

Linux/Alphaの結果は最内周をアセンブラで展開しているので少し有利になっています。
特筆するべきは21264の結果で、アセンブラの展開を使わなくても500MHzのプロセッサで
軽々と600MFLOPSの大台を越えてしまいました。

反対に情けないのがR10000を使って
いることになっている NEC UP4800/770ADでなんと100MHzの21064Aと大差ない
性能しかでていません。PentiumIIもなかなか性能を出しているので半端な
性能のワークステーションは生き残れないのではないでしょうか？
一つのベンチマークだけで評価するのは早計ですから、この結果でなにかを

述べるつもりはありませんが、それでも21264には大変期待が持てると言えるでしょう。

おわりに

Linux/Alphaに対するCOMPAQの積極的な姿勢が目立ってきました。
この雑誌が出るころにはさらに色々な発表が続いていると思います。

今回は間に合わないのですが、最新のプロセッサのベンチマークも
さらにいろいろな角度から行なうつもりで手筈を整えつつありますので、
ご期待ください。

********** コラム 2 **********
LSI CADのツールを載せるために私が個人で購入していたSUNのSS10の
ディスクがクラッシュしたので(OSメディアを持っていない私は)RedHat 5.2
を購入して来てLinuxを載せてしまいました。その後、やはりDebianに
統一したほうが便利なのでパーティションを分けてDebianを入れました。
SUNのフロッピーも普段使わないだけに信頼性に問題があるようで私の
マシンも全くフロッピーは読めませんでしたが、CDROMからブートした
RedHatをブートストラップとして無事Debianをインストールできました。
同じようなことをやる人は少ないかもしれませんが、参考までに手順を
示します。
1. HDをパーティション分割する。(このとき最初のパーティションは必ず
1GB以下にします。また、3番目のパーティションはwhole diskにして
おきます。
2. 最初のパーティションにRedHatをインストール
3. Debian用のパーティションと、RESCとRAMDISK用のパーティションを作成
4. RAMDISKパーティション(私の場合/dev/sda6)にRAMDISKイメージを解凍して展開
   zcat root1440.bin | dd of=/dev/sda6
5. resc1440.binのイメージをRESC用パーティション(/dev/sda5)に展開
   dd of=/dev/sda5 if=resc1440.bin
6. RESC用パーティションをマウントしてlinuxというファイルを/bootにコピー
7. /etc/silo.confを編集(私の場合次のようにした)

```
        timeout=50
        partition=1
        root=/dev/sda1
        image=/boot/linux
                label=debian
                root=/dev/sda4
                read-only
        image=/boot/vmlinuz-2.0.35-11
                label=linux
                root=/dev/sda1
                read-only
        image=/boot/linux
                label=ramdisk
                root=/dev/sda6
                read-only
```

8. siloを設定し、ramdiskのイメージをブート
(私の場合RedHatのsiloは何故かうまく動作せず、debianのbase systemの
siloをマウントして使った)
9. ramdiskのメッセージに従いdebianのインストール
10. debianの/bootはredhatの/bootへのシンボリックリンクに変更

以上で、SUN SS10へのDebianのインストールは無事済みました。
DebianのCDROMもなくフロッピーも使えなくても少し頑張れば
インストールできるのですね。
********** コラム 2 **********