Operating System Maniacs Version.2.0

ChorusOS
PS2Linux
EWS4800
VMWare

Far Northern Other World

ー Operating System Maniacs　Version.2 ー

お品書き

Jaluna-1/ChorusOS を動かしてみる

立神梢一

1.イントロダクション

Jaluna-1/ChorusOS とはなにか。

ChorusOS

http://www.experimentalstuff.com/Technologies/ChorusOS/

Jaluna-1(C5 カーネル)

http://sourceforge.net/projects/jaluna/

Jaluna 社

http://www.virtuallogix.com/

(2006/09/26 に、社名が Jaluna 社から VirtualLogix 社に変更されました)

Jaluna-1/ChorusOS とは、Chorus ソフトウェアが開発し、現在のライセンスは Sun Microsystems が所有しているマイクロカーネルの OS、Chorus を、ドイツの Jaluna 社(現 VirtualLogix 社)が、Jaluna-1 という OS としてリリースしているものです。
もともと SunMicrosystems の上記サイトからカーネルソースがダウンロードできるのですが、この ChorusOS を元にした Jaluna-1(C5 マイクロカーネル)も、sourceforge から、developers リリースとしてカーネルソースが、いわゆるフリーで公開されておりますので、その起動実験をして見たいと思います。
尚、実際のライセンスについては、実装部分ではロイヤリティは発生せず、パッチやサポート等の部分で収益を上げているようです。カーネルソースそのもののライセンスはまだ Sun にあるらしく、実際カーネルソースそのもののライセンスは MPL ライクな Sun のフリーライセンスに基づいての公開になっているようです。
また、VirtualLogix 社は、Jaluna-2/RT という OS がメインのコンポーネントのようで、これは上記のとおり ChorusOS の Version5 を、C5 カーネルとして、さらに Linux とのハイブリッドにした、ハードリアルタイム OS を指します。
本稿でテストを行うのは、そのうちの C5 カーネル(あるいは ChorusOS と呼ばれるもの)になります。Jaluna 社の実装では、これを Jaluna-1 と呼んでいるようです。

ChorusOS は、カラオケ「孫悟空」への搭載実績があるようですが、他はあまり存じあげません。ただ組込み OS 業界ではそれほどマイナーではないようです。
Installation Guide によると、(文章直訳)
「このオペレーティングシステムは公共のスイッチ、PBXs、アクセスネットワークと同じくらい良い十字接続スイッチ、ボイスメールシステム、セル基地局、ウェブ電話、および携帯電話の操作に使用されます。」
とのことで、組み込み制御向けの OS のようですが、
「UltraSPARC IIi/IIe、インテル x86/Pentium、モトローラパワーPC750/74×0 プロセッサファミリ、およびモトローラ PowerQUICC I(MPC8xx)、および PowerQUICC II(SBC8260)マイクロコントローラ」
をサポートしており、実装するならともかく、実験として行う分にはなんとでもなりそうです。

ちなみに CQ 出版者の、インターフェース増刊、「Embeded UNIX」Vol4 で、Jaluna-2 の特集が組まれたことがあり、紙面等で読める数少ない資料となっています。
(尚、蛇足もいいところですが、この「Embeded UNIX」の Vol2 では、LynxOS の特殊が組まれており、いずれ入手したい考えです。明らかに読者層とは違う気もしますが)
詳細なことについては、当方の説明よりも、Google 等で「Chorus OS」あるいは「Jaluna」「C5 kernel」などで調べたほうが手っ取り早いかもしれません。

2.必要なファイルの入手

上記 VirtualLogix 社のサイトからでも辿れますが、基本的には SourceForge から、Developpers Edition なるものがダウンロードできます。

Jaluna Real-Time Component Suite

http://sourceforge.net/projects/jaluna/

ドキュメントについては、当方の所持している C5 カーネル関連の殆どは、SunMicrosystems の Chorus/OpenSource や docs.sun.com からダウンロードできるものを改定しているもののようです。

VirtualLogix C5 - On-line Documentation

http://www.virtuallogix.com/index.php?id=46

のページからその殆どが閲覧可能ですし、詳細な説明がありますが、pdf 等でのダウンロードできる形状の物は無くなってしまったようです。

当方は、以前にダウンロードできたものを所持していますが、内容はまったく同じようです。

(といっても詳細に確認したワケではありませんが)

今回のテストは、この SourceForge のもソースと、以前 Jaluna 社からダウンロードしたドキュメントを用いて行うことにします。

3.事前準備

ファイルを入手したら、それ以外のハードウェアや環境等の準備を整えます。

ハードウェアとしては最低限以下の物が必要となります。

1.ChorusOS(jaluna-1)の Boot サーバになるマシン

2.ChorusOS(jaluna-1)を実際に起動するマシン

3.ChorusOS(jaluna-1)の動作を確認するのに必要なコンソールケーブル(クロス)

1.については、今回は Sparc マシンを使用します。ChorusOS は、Boot サーバとしては Solaris/Sparc しか使用できなかったようですが、Jaluna-1 では、Linux マシンを Boot サーバとして起動することも可能なようです。

今回は Boot 出来ることを最優先するので、動作する可能性がより高いと思われる Sparc マシンを使用します。

Ultra10(富士通 OEM/GPS7000MODEL10。CPU は差し替えて 300→440MHZ のクロックになっている)

UltraSparcIIi-440MHZ、MEM-512MB、HDD-30GB

2.については、いつも使っている VersaPro のノートマシンを使用します。ノートマシンではありますが、オンボードで Intel の NIC が載っているので、この手の実験については認識等でのトラブルが少ないと思われるからです。

K6II+-450MHZ、MEM256MB

3.についてですが、一般的なシリアルクロスのコンソールケーブル(Dsub9-Dsub9)を使用します。

ChorusOS(jaluna-1)は、いわゆる組込み系 OS だけあり、シリアルへの出力のみしか考慮されていないので、テスト的に Boot させる際には、シリアル経由でターミナルエミュレータに描画する必要があるからです。

事前準備として、ソフトとしては以下の物が必要となります。

1.Boot サーバ用の OS

今回は Boot サーバとしては Solaris10 を使用します。起動させることを優先すると書いていたのと矛盾するかもしれませんが、Install Guide によると Solaris8,9、or Higher となっているので、Solaris8 か 9 を使用するのが無難かもしれませんが、最新の OS で動かないと言うこともなかろうと思い、Solaris10 を使用しています。

Solaris については、ビルドに必要なライブラリ等に不足があることを防ぐためにフルインストールを行っています。無論 StarOffice なんかはいらないのでしょうけれど、面倒なので全部インストールしています。

勿論 CompanionCD も含めてインストールしています。

2.その各種設定

各項目の実行時に行っても良いのですが、事前に PATH 等の設定をしておきます。

GNU ツール群を使用できるように、作業用のユーザーの PATH 設定へ

/usr/local/bin

/usr/sfw/bin

/usr/ccs/bin

/usr/X/bin

を追加しておきます。

また、gmake や gcc を make や cc として認識するようにシンボリックリンクを貼っておきます。スクリプト内で make を実

行している箇所があるため、少なくとも gmka を make コマンドとして実行できないとうまく Build できないと判断したためですが、もしかすると当方の知識不足である可能性もありますので、詳しい方は別の解決方法もあるのかもしれません。今回はこれで上手くいったのでこの方法で行います。
/usr/ccs/bin を PATH に追加しているのは、ar を使用するためです。GNU の物を使ってもいいのかもしれませんが、今回はこの方法でうまく Build 出来たので、この方法で行います。
/usr/X/bin は、imake を使用できるように追加しています。(InstallGuide に記述あり)
その他各種ツールに関しても、いろいろと必要な前提条件があるとは思いますが、今回はこれで正常に Build 出来たために、特にライブラリ関係等あまり突っ込んで検証していません。一応、OS や各種代表ツールの Version 番号のみ記載します。
その他、各種 Daemon 等を起動する必要もあります。各項目で詳細に説明しますが、
rarpd
tftpd
NFS
が使用できる状態になっている必要があります。

OS 及びツールの Version

Solaris10:

```
# cat /etc/release
                        Solaris 10 3/05 s10_74L2a SPARC
    Copyright 2005 Sun Microsystems, Inc.  All Rights Reserved.
                        Use is subject to license terms.
                          Assembled 22 January 2005
```

gcc:

```
# gcc —version
gcc (GCC) 3.4.2
Copyright (C) 2004 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

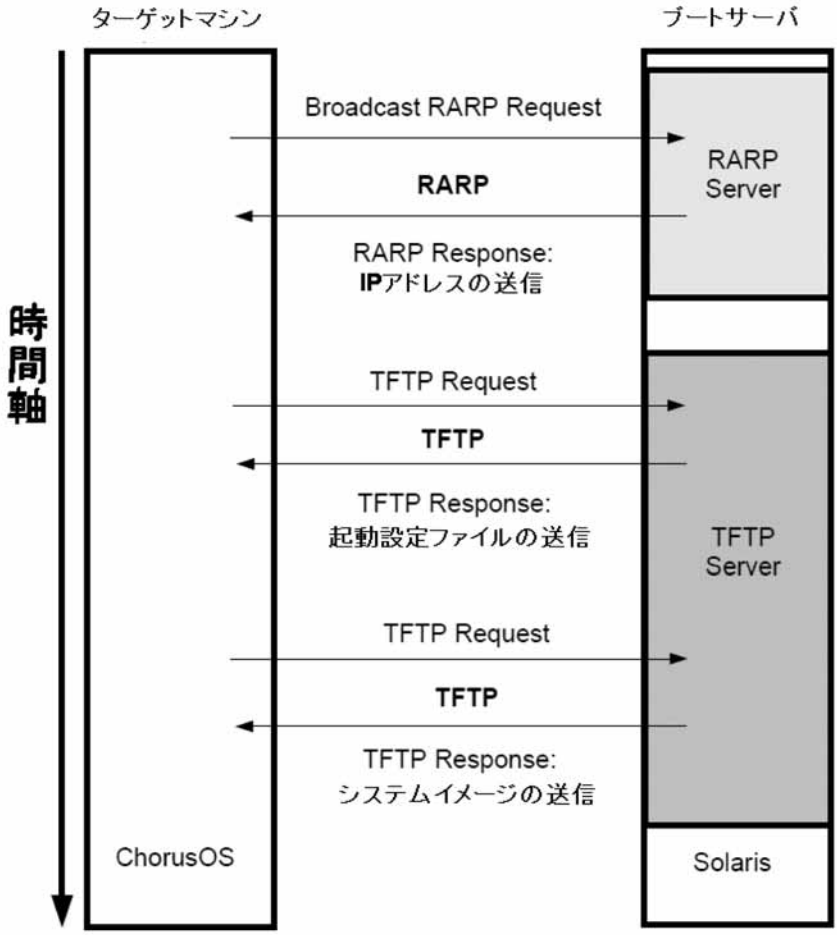
gmake:

```
# gmake —version
GNU Make 3.80
Copyright (C) 2002  Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
```

また、InstallGuide においては Boot サーバからシリアルコンソール接続し、起動画面を表示していましたが、当方は面倒を避けるため、ChorusOS(jaluna-1)を実際に起動するマシンへは、Windows マシンをコンソール接続し、TeraTerm にて起動状態の表示を行いました。
(よって、都合 3 台のマシンを使用しています)

4.インストール概要

Chorus はおおよそ次の動作を経て起動します。



その為に、まず Host 側への Chorus の導入を行います。

4-1.必要なファイル類

SourceForge の上記サイトから必要なファイルを入手してきます。
今回はあくまで「ChorusOS を起動する」ことのみを目標としますので、
c5-1.0-dev-sys-src.tgz
c5-1.0-dev-sys-x86.tgz
を使用します。

4-2.ファイルの展開

まず、上記のファイルを、サーバ側として使用するマシンの適当なディレクトリに展開します。

```
$gunzip < c5-1.0-dev-sys-src.tgz | tar xfv -
```

同様にもう一つのファイルも展開してしまってください。

インストールガイドにはオプションとして、クロスコンパイラのインストールと、gdb のインストールについての説明がありますが、今回はこれを行いません。

時間があればテストしてみたいと思いますが、当方は開発を行うわけではないので今回は申しわけありませんが省略します。

4-3.下準備

Host サーバは上記のとおり、Solaris/Sparc を使用します。異種アーキテクチャコレクションも当方の趣味の一つですが、実は大本を辿ると、そもそもが Chorus の実験がしたくて UltraSparc マシンを入手したのが最初でした。

異種アーキテクチャに傾倒するあまり本分を忘れていた?気もしますが、やっと念願の Chorus の実験に使われると言うことで、マシンも満足でしょう。

InstallGuide に沿って作業していきます。

1.Check /etc/nsswitch.conf on the boot server for lines starting with ethers:

/etc/nsswitch.conf の ehters の行がどのように記述されているかを確認し、files であれば問題なし。(Solaris10 をデフォルトでインストールしただけなのでそのようになっています)

2.Add the following line to /etc/ethers on the boot server

target_Ethernet_address target_hostname

/etc/ethers を、以下のように編集します。

target_Ethernet_address target_hostname

例)00:D0:59:0A:27:39 tindaros

問題なければ rarpd を起動しますが、とりあえずは設定をしておいて、後でまとめて起動することにします。

次に tftp サーバの設定をします。/tftpboot というディレクトリを作成します。ここに起動設定ファイルを置くことで、RARP サーバから IP アドレスを取得したターゲットマシンは、ChorusOS の起動イメージを tftp サーバから取得することが出来ます。

＊Solaris10 で RPRPD を起動するには:

svcadm コマンドで、rarpd を有効にします。これだけで動作するはずです。

```
# svcadm enable rarpd
```

＊Solaris10 で TFTP サーバを起動するには:

/etc/inetd.conf に

```
# tftp    dgram   udp6    wait    root    /usr/sbin/in.tftpd      in.tftpd -s /tftpboot
```

という行があるので、コメントアウトを外して

```
# inetconv
```

を実行します。すると TFTP サービスが SMF に組み込まれれるので、svcs コマンドを使って tftp/udp6 サービスが動いていることを確認します。

```
# svcs tftp/udp6
STATE          STIME    FMRI
online         21:09:21 svc:/network/tftp/udp6:default
```

4-4.OS の Build

これらの設定が済んだら、今度は起動イメージの Build に入ります。

まず、適当なところに Build 用ディレクトリを作ります。名称は任意で構いません。

尚、このディレクトリは NFS でマウント出来るように設定しておく必要があります。(起動後にターゲットマシンから NFS でマウントして使用できるようにするためです)

作成したディレクトリに移動します。

$ mkdir Build_Dir
$ cd Build_Dir
$ sh [[ファイルを展開したディレクトリ]]/tools-solaris/configure ¥
-f [[ファイルを展開したディレクトリ]]/src/nucleus/sys/x86/x86 ¥
-s [[ファイルを展開したディレクトリ]]/src/nucleus/bsp/x86/pc ¥
-s [[ファイルを展開したディレクトリ]]/src/nucleus/bsp ¥
-s [[ファイルを展開したディレクトリ]]/src/os ¥
-s [[ファイルを展開したディレクトリ]]/src/opt/X11
[[ファイルを展開したディレクトリ]]
は、上記でファイルを展開したディレクトリのパスです。Build_Dir からの相対パスでも絶対パスでも良いようです。
ちなみにこの Configure は、存在するオプションをかなりたくさん有効にした物ですが、この結果どのような機能が有効化されているのかまでは調査しておりません。英文ですがドキュメントは充実しているので、詳しく読み込んでいくともう少しいろいろ出来るのかもしれません。
Configure はすぐ終わりますので、続いて
$ make
をタイプします。これはかなり長い時間がかかります。
尚、ISA の Ethernet カードの場合は、build 時に値を埋め込んでおく必要があるようですが、まあいまどきのマシンで ISA ということもないと思うので省略します。
Make に成功したら、起動イメージの Build に入ります。
$ make chorus
とタイプすると、起動イメージの Build をはじめます。これもそれなりに時間がかかります。
しばらく待つと、最初に作った Build 用ディレクトリ、Build_Dir 内に、chorus.bmon と言う名前の SystemImage が作成されます。これが OS 本体の起動イメージになるようです。
続いて、NFS でマウントするための root システムイメージを作成します。
$ make root
Build_Dir 内にターゲットシステムからアクセスする各種バイナリファイルと設定ファイルが格納されます。

4-5 その他の設定の記述

まず、起動時に tftp サーバからターゲットマシンがシステムイメージを取得できるように、先ほど Build した起動イメージ、Chorus.bmon を、/tftpboot 直下に置きます。
$ cp chorus.bmon /tftpboot
また、そこに起動設定ファイルを置きます。
前号の SSS-PC の際にも、同様に/tftpboot に起動設定ファイルを置いていますが、これは tftp サーバから起動させる際の決まりごとなので、今後も同様の実験をする際に役立つかもしれません。
/tftpboot 直下に置く起動設定ファイルは、起動するターゲットマシンの IP アドレスを、Hex 表記にしたものを先頭に持つファイルを置きます。具体的には
[[Hex 表記の IP アドレス]].ChorusOS.5.0
というファイルになります。
Hex 表記というのは、お分かりの方もいらっしゃると思いますが、16 進数表記ということです。InstallGuide の物を引き合いに出して簡単に説明しますと、
ターゲットシステムの IP アドレスが 129.157.197.88 だとする。
. 129 十進数表記を十六進数表に直すと 81
. 157 十進数表記を十六進数表に直すと 9D
. 197 十進数表記を十六進数表に直すと C5
. 88 十進数表記を十六進数表に直すと 58
よって、/tftpboot 直下に置くファイル名は、
819DC558.ChorusOS.5.0
となる、ということです。
そして、このファイルの中に、起動用の設定を書いておきます。ブートサーバの IP アドレスが、129.157.197.144(上記の例にあわせてあります)だとすると、

AUTOBOOT=YES
BOOTFILE=chorus.bmon
BOOTSERVER=129.157.197.144
のような記載になります。
また、起動後に、ターゲットシステムから Chrous の root ディレクトリをマウントできるように、NFS サーバの設定をします。
/etc/dfs/dfstab に、以下のように root ディレクトリをマウントできるように設定を記述します。
share -F nfs /[[Build_dir までのフルパス]]/
[[Build_dir までのフルパス]]内に、そのままですが、Build_dir までのパスを記述します。
他のディレクトリに移動した上でそのディレクトリをマウントできるようにしても良いのかもしれませんが、試していません。また、セキュリティ上の理由等からすると、この設定ではいろいろと問題がある可能性があります。繰り返しになりますが、今回は起動させることを優先しての設定になりますので、ご注意下さい。
尚、記述後、再起動すると起動時に/etc/dfs/dfstab を読んで nfsd が自動で起動しますが、手動で起動させても良いはずです。

4-6.BootMonitor ディスクの作成

全ての設定をし終えたら、いよいよ起動用のディスクの作成に入ります。
まず、BootMonitor ディスク(起動設定用ディスク)の作成用ディレクトリを作ります。ここでは bootmon とします。

```
$ mkdir bootmon
$ cd bootmon
```

次に bootmon ディレクトリの環境を構成します。

```
$ sh install_dir/tools-host/configure ¥
-f source_dir/nucleus/sys/x86/x86 ¥
-s source_dir/nucleus/bsp/x86/pc ¥
-s source_dir/nucleus/bsp
```

これはすぐに終了するので、make コマンドで botmon 環境を構築します。

```
$ make
```

bootmon ディレクトリ配下の/conf/mini,が、テスト用の miniprofile になります。
内容は以下のようになっています。

```
#
# Mini Profile
#
#
# Kernel features
#
-set USER_MODE=false
-set POSIX_SHM=false
-set VIRTUAL_ADDRESS_SPACE=false
-set SEM=false
-set EVENT=false
-set MONITOR=false
-set TIMER=false
-set DATE=false
-set RTC=false
-set PERF=false
-set IPC=false
-set MIPC=false
-set LAPBIND=true
-set LAPSAFE=false
-set MON=false
-set LOG=false
```

-set DEBUG_SYSTEM=false

-set SOLARIS_SYSEVENT=false

InstallGuide 以外に、SysAdminGuide、という解説書があるのですが、こちらにこの手の起動時の設定等についてはかなり詳細に解説が載っています。開発系の知識のある方は、参考の上各種パラメタをいじるのも面白いかもしれません。当方もいくつか試しましたが、内容について詳細に追いかける余裕がなかったことと、起動優先のため、デフォルトの mini プロファイルにて Build します。

尚、basic や、extended といった別の設定ファイルも、同ディレクトリにあるようです。.

さて、BootMonitor ディスクの環境を構成します。

$ bin/cdstool configurator -p conf/mini

$ bin/cdstool configurator -set LOADER=lilo

$ bin/cdstool configurator -setenv BOOTCONF=RARP

問題なければ、特に出力は出ません。問題があると、警告メッセージのようなものが出ます。また、システムイメージの Build の際と同様に、ISA の Ether カードを使用している場合は、各種パラメタの埋め込みが必要なようですが、ここでは割愛します。

環境構成が終わったら、BootMonitor ディスクイメージを作成します。

$ make bootMonitor

しばらく時間がかかりますが、bootMonitor.imageという名称で、起動ディスクのイメージが作成されます。

生成された BootMonitor ディスクイメージを、フォーマットされたフロッピーディスクへコピーします。

rmformat -F quick /dev/rdiskette0

フォーマットを行うとディスク上のデータはすべて消去されます。

継続してもよろしいですか? (y/n)y

/etc/init.d/volmgt stop

cp bootMonitor.image /dev/fd0

/etc/init.d/volmgt start

*Solaris10 で行っていますので、オートマウントデーモンの停止、開始をしていますが、Linux の場合は関係ありません。

5.ディスクでのターゲットの Boot

さて、これで ChorusOS(Jaluna-1)起動のための準備が整いました。
いよいよ起動してみます。

5-1.クロスシリアルケーブルでの結線

ターゲットマシンのコンソールポートにクロスシリアルケーブルを接続します。
InstallGuide では*nix 系 OS での操作を前提にして記述されていましたが、当方では先にも書きましたとおり、面倒を避けるため、Windows マシンのシリアルポートへ接続し、TeraTerm にてログ採取を行いました。

5-2.ターゲットマシンの起動

BootMonitor ディスクでターゲットマシンを起動すると、ターゲットマシンのモニタには
Loading**********:
という文字列が表示されるだけで、他には何も出力されません。
コンソール側に、起動のログが表示されますので、その結果を貼っておきます。

```
dbgBsp initi
RAM size: 0xf800000 bytes
CPU: AMD-K6(tm)-III Processor (586-class CPU)
   Origin = "AuthenticAMD"   Id = 0x5d4   Stepping = 4AMD info

AMD info

C5 1.0 for Intel x86 - Intel x86 PC/AT
Copyright (c) 2002, Jaluna SA. All rights reserved.

Parts of the product are derived from ChorusOS 5.1, licensed from Sun
Microsystems under the Sun Public License Lite version 1.0.

Kernel modules : CORE SCHED_CLASS [ FIFO RR RT ] MEM_FLM KDB TICK ENV BLACKBOX MUTEX PERF TIMEOUT LAPBIND DKI
/pci/i8259: sun:pci-i8259-pic driver started
/pci: sun:x86-bios-(bus,pci,mngt) driver started
/pci/pci-isa: sun:pci-bios-(bus,isa) driver started
/pci/pci-isa/i8254: sun:bus-i8254-timer driver started
/pci/pci1106,501@0,0: device node is created by sun:pci-enumerator-
/pci/pci1106,8501@1,0: device node is created by sun:pci-enumerator-
/pci/pci8086,1229@6,0: device node is created by sun:pci-enumerator-
/pci/pci1106,686@7,0: device node is created by sun:pci-enumerator-
/pci/pci1106,571@7,1: device node is created by sun:pci-enumerator-
/pci/pci1106,3038@7,2: device node is created by sun:pci-enumerator-
/pci/pci104c,ac1c@a,0: device node is created by sun:pci-enumerator-
/pci/pci104c,ac1c@a,1: device node is created by sun:pci-enumerator-
/pci/pci8086,1229@6,0: Phy detection — 100 Mbit/s Full Duplex Link
/pci/pci8086,1229@6,0: Individual Address 00:d0:59:0a:26:c6
/pci/pci8086,1229@6,0: driver sun:pci-e100-(ether,mngt) started
TICK: using timer device /pci/pci-isa/i8254/i8254, at 100 Hz

Boot Monitor Loader v1.3 (env BOOTCONF)

env BOOTCONF: 'RARP'
*** booting using 'RARP' agent
Unit: 0   device name: pci8086,1229@6,0

Using unit 0
```

～～～長いので省略していますが、ここの上までの部分が、2 回繰り返して表示されます。BootMonitor ディスクでの表示と、tftp で取得したイメージでの表示の 2 回と思われます～～～～

```
My IP 192.168.1.40, RARP Server IP 192.168.1.50

Loading file C0A80128.ChorusOS.5.0 on server 192.168.1.50: loaded!

Loading file chorus.bmon on server 192.168.1.50: | 0000100/ 0000200- 0000300¥ 0000400| 0000500/ 0000600- 0000700¥ 0000800| 0000900/
0001000- 0001100¥ 0001200| 0001300/ 0001400- 0001500¥ 0001600| 0001700/ 0001800- 0001900¥ 0002000| 0002100/ 0002200-
0002300¥ 0002400| 0002500/ 0002600- 0002700¥ 0002800| 0002900/ 0003000- 0003100¥ 0003200| 0003300/ 0003400- 0003500¥ 0003600|
0003700/ 0003800- 0003900¥ 0004000| 0004100/ 0004200- 0004300¥ 0004400| 0004500/ 0004600- 0004700loaded!
Boot new system image ...
```

```
dbgBsp initiDebugAgent: trying to sync with DebugServer... No response.

RAM size: 0xf800000 bytes
CPU: AMD-K6(tm)-III Processor (586-class CPU)
  Origin = "AuthenticAMD"  Id = 0x5d4  Stepping = 4AMD info

AMD info

C5 1.0 for Intel x86 - Intel x86 PC/AT
Copyright (c) 2002, Jaluna SA. All rights reserved.

Parts of the product are derived from ChorusOS 5.1, licensed from Sun
Microsystems under the Sun Public License Lite version 1.0.

Kernel modules : CORE SCHED_CLASS [ FIFO RR RT ] SEM MIPC IPC_L MEM_PRM KDB TICK MON ENV ETIMER LOG BLACKBOX
LAPSAFE MUTEX EVENT UI DATE PERF TIMEOUT LAPBIND DK
I
MEM: memory device 'sys_bank' vaddr 0x7bb7a000 size 0x252000
/pci/i8259: sun:pci-i8259-pic driver started
/pci: sun:x86-bios-(bus,pci,mngt) driver started
/pci/pci-isa: sun:pci-bios-(bus,isa) driver started
/pci/pci-isa/i8254: sun:bus-i8254-timer driver started
/pci/pci-isa/mc146818: sun:bus-mc146818-(rtc,timer) driver started
/pci/pci-isa/ns16550-2: sun:bus-ns16550-uart driver started
/pci/pci1106,501@0,0: device node is created by sun:pci-enumerator-
/pci/pci1106,8501@1,0: device node is created by sun:pci-enumerator-
/pci/pci8086,1229@6,0: device node is created by sun:pci-enumerator-
/pci/pci1106,686@7,0: device node is created by sun:pci-enumerator-
/pci/pci1106,571@7,1: device node is created by sun:pci-enumerator-
/pci/pci1106,3038@7,2: device node is created by sun:pci-enumerator-
/pci/pci104c,ac1c@a,0: device node is created by sun:pci-enumerator-
/pci/pci104c,ac1c@a,1: device node is created by sun:pci-enumerator-
/pci/pci8086,1229@6,0: Phy detection -- 100 Mbit/s Full Duplex Link
/pci/pci8086,1229@6,0: Individual Address 00:d0:59:0a:26:c6
/pci/pci8086,1229@6,0: driver sun:pci-e100-(ether,mngt) started
DATE: using rtc device /pci/pci-isa/mc146818
TICK: using timer device /pci/pci-isa/i8254/i8254, at 100 Hz
C_OS: Copyright 1994-2002 FreeBSD, Inc. All rights reserved.
C_OS: FreeBSD 4.1-RELEASE
C_OS: Current date: Sat Oct  14 01:30:09 GMT 2006
DISK: warning -- svDeviceLookup(disk) failed (-7)
CDROM: warning -- svDeviceLookup(cdrom) failed (-7)
/rd: sun:ram--disk driver started
C_OS warning: Initialization of DEV_NVRAM feature fails
C_INIT: console started
C_INIT: /image/sys_bank mounted on /dev/bd00
C_INIT: found /image/sys_bank/sysadm.ini
C_INIT: executing start-up file /image/sys_bank/sysadm.ini
C_OS warning: IfEth 7fdef8c8 - Ethernet link UP
C_OS: ifeth0 bound to device /pci/pci8086,1229@6,0
ifeth0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 192.168.1.40 netmask 0xffffff00 broadcast 192.168.1.255
        ether 00:d0:59:0a:26:c6
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet 127.0.0.1 netmask 0xff000000
C_INIT: rshd started
C_INIT: init in unsecured mode
```

この状態で、ターゲットマシンで ChorusOS が起動し、rsh でのコマンド受付状態になります。
尚、起動ログをご覧になると記述があるとおり、ChorusOS においては、Posix 互換部分は FreeBSD を利用しています。

6.いくつかのコマンドを実行してみる

tindaros というのが ChorusOS を起動しているターゲットマシンの Hots 名です。
IP アドレスは、Solaris が動いている Chorus ブートサーバが 192.168.1.50、ターゲットマシンが 192.168.1.40 です。
たいしたことはしていません。一般的なコマンドを叩いてみただけです。実際にもうすこし開発系の知識があれば、IO を叩くとかいろいろ遊べるのかもしれませんが。。。
この手の OS においての、初歩的な、あるいは面白い実験の方法などご存知の方、是非ご教授下さい。

```
# rsh tindaros help
C_INIT C5 1.0.0-RELEASE- valid commands that deal with:

File Systems:
        mount [[-dfruvw] [-o options] [-t nfs|ufs|msdosfs|pdevfs] host:pathname|special_file [mount_point]]
        umount [-v|-F|-f|-a|-t nfs|ufs|msdosfs|pdevfs] [special_file]
        swapon [mount_point]

Actors:
        arun [-g rgid] [-S | -U] [-k] [-T] [-d] [-q] [-D] [-Z] [-xip] path [args]
        akill [-s site] [-g rgid | [-c] aid ]
        pdump   aid
        aps
        umask [mode]
        ulimit [-HSafn] [limit]

Environment variables:
        setenv var value
        unsetenv var
        env

Networks:
        route
        netstat
        ping host
        ifconfig
        ifwait ifname [timeout, default infinite]
        rarp ethernet_interface_name
        pppd
        pppclose device
        pppstop
        ethIpcStackAttach [dtreepath]

Devices:
        mknod name [b | c] major minor
        dtree
        mkdev name unit [dtreepath]

This Target:
        reboot
        restart
        memstat

This shell:
        echo string
        source filename
        cd [directory]
        sleep [time in seconds, default=1s]
        help
        console
        rshd
```

ヘルプです。おおよそのコマンドはここで確認できます。なお、BootMonitor の設定オプションによって、組み込まれないものもあるようです。

```
# rsh tindaros netstat
started pid = 6
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  192.168.1.40.1022      192.168.1.50.1018      ESTABLISHED
tcp4       0      0  192.168.1.40.514       192.168.1.50.1019      ESTABLISHED
tcp4       0      0  192.168.1.40.1023      192.168.1.50.1020      TIME_WAIT
```

```
tcp4       0      0  192.168.1.40.514       192.168.1.50.1021      TIME_WAIT
# rsh tindaros env
PATH=/bin:/sbin
HOST=
TZ=GMT0
# rsh tindaros cd
/
# rsh tindaros mount 192.168.1.50:/root/Chorus/jaluna/test/chorus/root /
started pid = 8
```

NFS のパスがひどい有様なのは、まあ起動優先の環境と言うことでご容赦下さい。NFS マウントすると、Solaris 側の/root ディレクトリにアクセスが可能となります。多少、コマンド等出来ることも増えるのかもしれません。また、クロスコンパイル環境を整えた場合はおそらくこの状態でコンパイルを行うことになると思われます。

```
# rsh tindaros mount
started pid = 9
root_device on / (pdevfs)
pdevfs on /dev (pdevfs, local)
pdevfs on /image (pdevfs, local)
/dev/bd00 on /image/sys_bank (msdos, local, read-only, reads: sync 14 async 758)
192.168.1.50:/root/Chorus/jaluna/test/chorus/root on / (nfs)
# rsh tindaros ls
started pid = 10
Makefile        etc             proc            usr
bin             image           sbin            var
dev             lib             tmp
# rsh tindaros cd usr
/usr
# rsh tindaros ls
started pid = 11
lib
# rsh tindaros ls /etc
started pid = 12
disktab         group           protocols       services
exports         master.passwd   resolv.conf     syslog.conf
fstab           netconfig       sample
# rsh tindaros ls /var
started pid = 13
db      log     run
# rsh tindaros cd /var/log
/var/log
# rsh tindaros ls
started pid = 14
Makefile        etc             proc            usr
bin             image           sbin            var
dev             lib             tmp
# rsh tindaros ls /bin
started pid = 15
PROF            df              inetNShost      netstat         sleep
arp             disklabel       inetNSien116    newfs           swapon
bootconfig      domainname      inetNSnis       newfs_msdos     sync
cat             fdisk           kill            nfsstat         sysctl
cbfs            flashdefrag     logger          pax             tclsh
cdcontrol       format          ls              ping            tftp
chmod           fsck            mkdir           ping6           touch
chorusNSinet    fsck_dos        mkfifo          pppstart        traceroute
chorusNSsite    fsck_msdos      mknod           pwd_mkdb        umount
chorusStat      ftp             mount           rm              uname
cp              gifconfig       mount_msdos     rmdir           unloadDrv
cs              hostname        mount_nfs       route           ypcat
date            ifconfig        mv              rpcinfo         ypmatch
dd              inetNSdns       ndp             shutdown        ypwhich
```

ディレクトリを順番に見ているだけです。

```
# rsh tindaros df -h
started pid = 16
Filesystem                                      Size    Used   Avail Capacity  Mounted on
root_device                                     16K     16K      0B    100%    /
pdevfs                                          16K     16K      0B    100%    /dev
pdevfs                                          16K     16K      0B    100%    /image
```

```
/dev/bd00                                                    7.0M    2.3M    4.7M     33%    /image/sys_bank
192.168.1.50:/root/Chorus/jaluna/test/chorus/root    32G    6.8G     25G     22%    /
# rsh tindaros ping 192.168.0.1
C_INIT: ping: sendto: No route to host
# rsh tindaros ping 192.168.1.50
192.168.1.50 is alive
# rsh tindaros ping 192.168.1.1
192.168.1.1 is alive
```

Ping を打ってみました。ゲートウェイが設定されていないため、別サブネットにはルートが無いと言われています。ゲートウェイやルーティングも設定出来るようですが、今回はそこまで検証していません。すいません。

```
# rsh tindaros ls /etc
started pid = 17
disktab          group            protocols        services
exports          master.passwd    resolv.conf      syslog.conf
fstab            netconfig        sample
# rsh tindaros cat /etc/resolv.conf
started pid = 18
domain fnow.org
nameserver 192.168.0.126
search fnow.local
# rsh tindaros ls -la /
started pid = 19
total 19
drwxr-xr-x   12 0   wheel    512 Oct   9 15:59 .
drwxr-xr-x   12 0   wheel    512 Oct   9 15:59 ..
-rw-r--r--    1 0   wheel   7180 Oct   9 15:59 Makefile
drwxr-xr-x    2 0   wheel   1536 Oct   9 16:00 bin
drwxr-xr-x    1 0   wheel    512 Jan   1   1970 dev
drwxr-xr-x    3 0   wheel    512 Oct   9 15:59 etc
drwxr-xr-x    1 0   wheel    512 Jan   1   1970 image
drwxr-xr-x    3 0   wheel    512 Oct   9 15:59 lib
drwxr-xr-x    2 0   wheel    512 Oct   9 15:59 proc
drwxr-xr-x    2 0   wheel    512 Oct   9 16:00 sbin
drwxrwxrwx    2 0   wheel    512 Oct   9 15:59 tmp
drwxr-xr-x    3 0   wheel    512 Oct   9 15:59 usr
drwxr-xr-x    5 0   wheel    512 Oct   9 15:59 var
# rsh tindaros shutdown
started pid = 20
```

Shutdown と reboot のコマンドが入力されると、Chorus は再起動します。(Shutdown でもマシン自体は再起動します)
また、このコマンドが入力されると、ターゲットマシンに繋いだコンソールには、以下のような表示が出力され、ターゲットマシンは再起動します。

```
shutdown in progress...
Cold reboot ...
```

以上が ChorusOS の Build～起動と簡単な動作確認の記録になります。

7.おわりに

今回は、OS マニアをはじめてから、ずっと試してみたいと思っていた ChorusOS の実験を終えることができ、とても満足しています。
ただ、実際にこうやって実験をすると、今度は組込み系や、プログラミング系の知識が無いことでかなり歯がゆく思いました。
ChorusOS も、BotMonitor ディスクの設定時にいろいろとオプションを調整することで CD-ROM ドライブやローカルディスクへのアクセスが可能だったり、クロスコンパイル環境を整え、簡易な実験プログラムなどを動かすことで、もっと充実したテストも出来たのでは無いかとの反省もあります。
こういったテストは以降も継続して行い、いろいろと面白い実験が出来ればと思います。

ChorusOS を触って思ったのは、ドキュメントも充実していますし、それなりに組込み系プログラミングの知識のある方なら、もっと有用に使うことも出来るのかな、といったことでした。上記に書いたこととも若干被りますが、リソースが少なくても動作はするようですし、ネットワークも周辺記憶デバイスも扱えますから、何か面白いことも出来そうです。それが何なのかをハッキリわからないあたりが当方の限界かとも思いますが。
また、今回は実験の成功を重視してターゲットマシンに PC を、Boot サーバに Solaris を使用しましたが、Linux を Boot サーバにも出来るようですし、ターゲットマシン(正確には CPU か)も、かなりいろいろなアーキテクチャに対応していますので、また記事として別途まとめられたら嬉しいです。

ChorusOS というか、今回の Jaluna-1 は、発売元が社名を変更してはいますが、まだ SourceForge からダウンロードすることも出来ますので、実験好きな方、そして何より組込み系エンジニアの方、是非試してみていただければと思います。そして、なにか面白い用途や実験などありましたら、ネタ提供いただければ当方でも追加でどんどんと実験をしていきたいと思っています。

<了>

NEC 国産 OS への鎮魂曲

りろ@涅槃

1 それは1993年の冬だった

一介の事務屋に過ぎなかった私に下された命令。それは「重役向けのデータベース検索システムを作れ」であった。それまで自分は NEC の汎用機 ACOS で動作する業務システムの運用を任されていたが、システム開発の経験など全くなかった。その私になぜそんなご無体な命令が下されたかというと、当時コンピュータを扱える人間がほかにいなかったからだ。件のデータベース検索システムは一応社内事情なので詳細は伏せさせていただくが、気まぐれな重役のアドホックな資料要求に課員全員がすっかり疲弊した挙句、「重役にシステムを預けてしまえ」ということになった次第である。
追い風としてあったのは、ダウンサイジングブームとオープン化である。馬鹿でかくて融通の利かない、しかも人的金銭的負担の大きい汎用機から、UNIX サーバを用いたクライアント／サーバシステムへの移行は社内的にも歓迎の対象だったのだ。

とはいえ現在のようにパッケージを探してきてカスタマイズするという時代ではない。全部一からの構築である。出入りのシステムインテグレータに話を振ったところ、1週間後、血走った眼をした営業さんが集団で押しかけて、分厚い提案書を持ってきた。1週間不眠不休で作り上げたらしい。ご苦労なことである。提案の内容は、現在運用中の ACOS 業務システムに改良を加え必要な情報を入力させたあと、ISAM 形式ファイルを RDB 化して当時最先端だったエンジニアリング・ワーク・ステーション NEC EWS4800 24台に配信するというものだ。検索システムは EWS4800 上で動作する X/Motif アプリケーションを開発する。

費用面で考えれば、24台の EWS 配備は無謀というもので、サーバ1台に X 端末を接続するのが筋というものだ。しかし当時のネットワーク環境は ISDN 10BASE-T だ。サーバのスループット等も考慮すると、同一社屋内ならともかく、12支社からの X 端末接続ではとうてい実用に耐えないだろうというベンダーの主張であった。もちろん沢山ハコモノを売りたいベンダーのスケベ心ではあるが一応正論だ。

かくして、私が初めて経験するシステム開発でおつきあいすることになった OS は、NEC ACOS6 NEC N5200(PTOS) NEC EWS4800(UX/4800)であった。 ちなみに RDBMS は ORACLE である。

2 各 OS の歴史(Wikipedia から引用)

NEC EWS4800(UX/4800)

CAD/CAM 分野で使用されているグラフィックに特化したワークステーションにおいて、1980 年代後半から 2000 年頃までは UNIX を採用したものが多かった。
NEC の EWS も例に漏れず自社製 UNIX を使用し、1993 年頃までトップシェアを誇っていた 2 次元 CAD ソフトの CAE-2D/MA[E]及び、SX-OS(ACOS4 改造版)から移植した 3 次元 CAD ソフトの CAE-3D/MA[E]及び、地図管理ソフト MAPVIEW をメインにシェアを伸ばしていた。
古舘伊知郎による「四方八方、4800 シリーズ」などのコマーシャルもあり、郵政省や通産省などの主要官庁系システムにも採用され、「NEC のオープン化の旗印」とされていた。そのため、国産 UNIX といえば NEC の UNIX という時期もあった程である。
ハードウェア自体も、1990 年初頭に当時主流だった CISC 機から RISC 機に切り替わり、RISC への移行が行われた。当時、SONY 及び住友電工、DEC などと MIPS アーキテクチャの CPU によるバイナリ互換協定を結び、さらにシェアを広げようとしていたが、途中で協定の主役の一人である DEC が Alpha チップを発表し、脱落した。
さらに、1993 年頃からの UNIX 戦争とそれに続く商用 UNIX 間の生存競争が激しくなり、NEC の UNIX サーバ用 OS であった UP-UX との統合が図られた。開発リソースの集約にて UX/4800 という名称に変更されリリースが続けられた。しかし、次第に先細りになる国産 UNIX のシェアと自社サーバと自社 OS の開発費増大に NEC 自体が音を上げ、HP の OEM 化軍門に下ったため、急速に採用数が減っていき、現在はほぼ絶滅状態である。
ただし、この OS で育ったエンジニアがオープンソースプロジェクトにて Ruby や Struts などの開発で主力を務めており、国内のソフトウェア産業に与えた影響は大きい。

NEC ACOS6

1950 年代のパラメトロンコンピュータや FONTAC(富士通/沖/NEC 共同コンピュータ)の流れの中、1960 年代半ばに始まる通産省主体の大型プロジェクト超高性能電子計算機開発計画において、IBMなどの海外のコンピュータベンダに寄らない日本独自のコンピュータシステムを構築すべく、NEC/日立製作所/富士通/東芝/沖電気/松下電器/三菱電機などに通産省の元でコンピュータシステムの開発を進めさせた。松下などは、コンピュータはまだ商売にならないと判断し、早々と撤退を行ったが、それ以外のベンダはコンピュータの開発に鎬を削っていく事になる。
その後、日本では 1973 年に米国からの圧力などでコンピュータの輸入自由化が決定された。通商産業省は、当時の国内コンピュータメーカーの体力では IBM を初めとする海外メーカーに日本市場を席巻され打撃を受けるとして、当時 6 社あったコンピュータ業界の再編に乗り出した。富士通と日立、三菱電機と沖電気、それに東芝と NEC の 3 グループにまとめ、技術研究組合を作らせて 5 年間にわたって補助金を支給し、各社に「IBM 対抗機」の開発に当たらせた。
当時 NEC がハネウェル社から、東芝がジェネラル・エレクトリック社 (GE) から、それぞれ技術供与を受けてコンピュータを開発していた。ところが、1971 年に GE はコンピュータ事業から撤退して事業をハネウェルに売却し、ハネウェルは統合したコンピュータ部門をハネウェル・インフォメーション・システムズ社 (HIS) として独立させた。このため、同社と提携する NEC と東芝がグループを組むことになった。
両社は共同開発にあたり、小型機・中型機を NEC で、また大型機を東芝で、それぞれ開発を分担した。このため当初は中型機用として開発された ACOS-4 と大型機用の ACOS-6 はかなり異なるアーキテクチャのシステムとなった

NEC N5200(PTOS)

N5200(エヌゴーニーマルマル)は 1981 年に NEC が発売を開始したパソコンのシリーズ名である。ビジネスパソコンに分類されることもある。オフコンやワークステーションに分類されることもある。
通常はスタンドアロン利用のためのアプリケーションソフトも用意され、ローカルのハードディスクをデータストアとして利用できた単独で使用される。
通信系のオプションを装着することにより ACOS 系メインフレームの端末として機能したほか、
N6300/N6500/S3050/S3100S モデル系のオフィスプロセッサのターミナルコントローラとしても機能した。
OS は PTOS というものである。PTOS については、後に PC-PTOS という PC-9800 シリーズのハードウェア上で動くものも発売された。また、N5200 上で動作する MS-DOS も発売されている。但し、N5200 上で利用可能な全てのハードウェア資源は使えなかった。使用されるフロッピィディスクなどのメディアに於いては大型コンピューターの ACOS シリーズなどと互換性を持っていた。
派生系としてはワードプロセッサに特化した文豪シリーズがある。
元々、N5200 は PC-9800 シリーズと非常に似通ったハードウェアアーキテクチャを採用していた。CPU も 80x86 系、メモリの割り当てや IO チップもほぼ同じ(ポートもほぼ同じ)。VRAM 部分はどちらかというとハイレゾ系の PC-98 と似通っていた。当時、NEC はパーソナルユースとビジネスユースにて異なる事業展開を行っていた。この為、N5200 シリーズが大型コンピューター系の事業部、PC-9800 シリーズがパーソナルコンピューター系の事業部で事業展開を行っていた。
表計算ソフト『LANPLAN』(Microsoft 社の MultiPlan のもじり)、ワープロソフト『LANWORD』、データベースソフト『LANFILE』、グラフ作成ソフト『LANGRAPH』などのビジネスデスクトップアプリケーションソフトが用意された。
ライバルは富士通の K-10 および FACOM9450/FM-G シリーズであった。こちらの陣営も LAN シリーズ対抗製品として EPOC シリーズというビジネスデスクトップアプリケーションを発売していた。また、日立の 2020 や IBM のマルチステーション 5550 もライバル機種の一つであった。

3 かくしてシステムは完成した

自分の職務は、要件定義と画面遷移設計である。単純といえば単純だがお手本が何もないのだから、徹夜の連続が続いた。当時は用語すら知らなかったがウォーターフォール型開発では要件定義フェーズにおいて仕様凍結しなければ、開発を始められない。ベンダーからの矢の催促を浴びながら、満身創痍で仕様は確定した。一度仕様が確定すれば、あとは仕様書の細かい疑問点について解決していけばよい。
システムの構成と処理手順を簡単に説明すると、
(1)事務担当者が日次データを N5200 端末からバッチ処理で入力する
(2)エラーチェック処理とトランザクションは ACOS6 側のシステムで行う
(3)EWS4800 が、これを吸い上げ RDB として蓄積する。なおこのときの通信プロトコルは DYNA であった
(4)本社 EWS4800 が ISDN 回線を通して日々差分を支社の EWS4800 に配信する(レプリケーション)
(5)単純な検索集計は、オリジナル開発のメニューを用い、複雑な集計はこれも NEC のアプリケーション Micro Reseacher を使用する

カットオーバーは1994年の4月であったが、そのときの重役からの評判は散々であった。

(1)操作が難しい(というかコンピュータに無縁の人たちなのだからそもそも無理だったのだ)
(2)音がうるさくて暑い(2Gの外付けハードディスク6基を装填していた。確かにうるさくて暑い)
(3)欲しい帳票が手に入らない(自分のイメージ先行型の人は、汎用帳票を認めない)

そのほか定期的にヒアリングを繰り返した結果、膨大なバックログを抱えて、システムは厄介なお荷物と化していく。まず一番の問題は重役自身が自ら操作する気が全くないので、秘書や担当課員が重役室にこもって作業請負をするという結果になったことである。同時に、同年リリースされたマイクロソフトの Windows95 によってパソコンの操作性が飛躍的に向上したことによって、一気に EWS4800 の旗色が悪くなった。
1997年頃には Microsoft Office の性能が安定し、巨大な EWS4800 は単にデータストアとしての意味しか持たなくなった。枢要データを切り出したシーケンシャルな CSV ファイルを取り出して Windows(MS OFFICE)上で加工することが多くなった。そのため EWS4800 は、ますます邪魔な存在となっていった。EWS4800 でのフロントエンドは X/Motif アプリケーションであったため、Windows95 に PC-X サーバを実装して操作する案も出て、実際に実験が行われたが、やはり当時のネットワークスループットでは実用に耐えなかった。

ふたたび膨大な予算をかけて、Windows 上で動作するネイティブなフロントエンドアプリケーションを開発し、EWS4800 は暗い端末室にお引っ越しとなった。そのまま小規模な改良が毎年のように繰り返された。さらに使い勝手が悪いという支社は、EWS4800 のデータを活用した独自の帳票出力システムを開発したり、独自アドオンを付け加えたりで、魔境と化していった。そしてついに 2004 年頃から全社最適を図るシステム刷新が行われ、個々のシステムは原則としてメニュー管理プログラムに登録して使うことになった。かくして EWS4800 で稼働するシステムはその役割を終え、カスタマイズされたパッケージソフトが導入(Solaris 上で稼働)されることになった。2006 年 10 月のことである。

4 NEC 国産 OS たちへのレクイエム

1990 年代前半、ルーツはアメリカ産であったとしても、とりあえず国産 OS は元気だったのだ。弊社では国産 UNIX といえば、EWS4800（UX/4800） オフコンなら PTOS だったのだ。歴史的には独自アーキテクチャ百花繚乱の時代であった。HP-UX AIX SunOS DigitalUNIX VMS と、それらに対応するプロセッサ MIPS Alpha SPARC PA-RISC 等。どれもみな憧れの対象であった。今でも筆者のあこがれの的は UNIX ワークステーションである。

かつて一世を風靡した NEC 国産 OS たちはどこへ逝ったのだろう。流通上は既に絶滅しているのは間違いない。ACOS だけは ITANIUM 汎用機でパラレル ACOS として生き残っているが繁栄は望めない。自分にとっては PTOS こそが仕事の友であった。汎用機または UNIX サーバから切り取ったファイルを LANFILE で集計し、LANPLAN/Graph で図表化するのが仕事であった。既に自分が現役を退いた 2000 年に赴任した事務所には、馬鹿でかい NEC N5200 が鎮座していた。何に使うのかというと、旅費支給の金種別計算に使うだけだそうだ。今では Excel と Word でほとんどの用事が済んでしまう

弊社の現在の各システムのサーバ OS で圧倒的シェアを誇るのは Solaris である。業務系システムでは WindowsNT サーバもまだまだ現役だ。今後は Windows Server2003 などにリプレースされていくだろうが。そして思い出の EWS4800 は 2000 年に出荷停止となった。

今はアーキテクチャといえば x86、OS といえば Windows または Solaris である。エンタープライズ環境では AIX や HP-UX をたまに見かける程度だ。Linux はそのスケーラビリティにもかかわらず、どちらかといえば、エンベディッド方面での活躍の方が目立つように思う。これからは CELL プロセッサの躍進に期待している。

90年代は遠くなりにけり。国産 OS の一世風靡も今は昔。それでも、かつて仕事で世話になった恩義は忘れない。秋葉原のジャンク屋で見かけたら、優しい言葉をかけてやりたい。ありがとう国産 NEC OS。

PS2 Linux をいじってみる

立神梢一

1.イントロダクション

PS2 Linux とはなにか

PS2

http://www.jp.playstation.com/

PS2 Linux オフィシャルページ

http://www.ps2linux.com/

PS2 Linux とは、その名のとおり、「PS2 の上で稼動する Linux」です。

以下は WikiPedia より引用――――――――――――――――――――――――――――

もともと、PS2 の開発環境は Linux ベースであることが知られており（ただし、開発機材に接続される PC が Linux ベースであるというだけで、よく誤解されていたように PS2 で Linux が動いていたわけではない）、「SCEI は Linux に積極的なようだ」「PS2 でも Linux が動くのではないか」という期待が一部で高まっていた。

そのような状況で、SCEI の久夛良木健社長が、日本 Linux 協会の生越昌己会長に対し「(PS2 で動く Linux は）出そうと思えば明日にも出せる」と発言したことから、ネット上で「PlayStation2 で動作する Linux の発売を求める署名運動」が開始された。2001 年 3 月 4 日のことである。

これを受け、同年 4 月 26 日、SCEI から PS2 Linux の発売が公式に発表された。5 月 9 日にベータ版の予約注文の受付が始まり、7 月ごろ発送された。後に、2002 年 1 月 30 日に正式版(Release 1.0)が発表された。

――――――――――――――――――――――――――――以上、WikiPedia より引用

さて、上記のとおり、メーカー純正のディストリビューションとして発売された PS2Linux ですが、全盛期は短いものでした。理由は上記で一部引用させていただいた WikiPedia にも載っていますが、一言で言えば、

「思ったほど高性能じゃなかったから」

ということに集約されると思います。厳密に言えば PS2 の高性能を引き出せる、引き出すプログラミングを行うのは不可能、もしくはあまりに敷居が高すぎたから、でしょうか。

ですが、当方は OS マニアであり、OS コレクター、異種アーキテクチャマニアでもあります。今回、相方の好意によってようやく入手した、PS2Linux で、いくつかの実験を試みたいと思います。

今回試みる実験の最終目標は、本来、DVD-ROM を使用して Boot する必要のある PS2Linux を、DVD-ROM なしに起動してみようというものです。

2.用意したもの

オリジナルの PS2Linux パッケージは、HDD ユニット、イーサネットアダプタ、USB キーボード、マウス、VGA ケーブルセット、これにインストールメディアが加わったものになるようです。

当方が入手したものは、BB ナビゲータ(以下 BBN と表記)が 2.0 のものでした。オリジナルでは HDD フォーマット用プログラムがあったようですが(未確認)、とりあえず当方の環境に合わせて記述します。

また、今回の実験では、PS2Linux とゲーム用の環境の共存インストール実験を行いますが、この際に別途 Linux が稼動している PC が必要になります。

尚、これは netcat、zcat、gzip 等のプログラムが揃っていれば、おそらくその他の*NIX 系 OS でも作業は行えると考えていますが、今回は検証していません。

さらに、各種実験を行う上で、β 版と 1.0 では若干違う部分があるようです。今回は 1.0 版を使用しての実験となります。1.0 とβの最大の違いは、β の場合、はじめから意識せずとも共存インストールが行える点にあります。ただし、アップデートなどの作業においては、1.0 のほうがやりやすい部分もあるようです。

使用した機器一覧

Linux インストールキット

HDD ユニット：BBN もしくは Linux で起動しないと HDD を認識しません。

イーサネットアダプタ(兼、HDD インターフェース)：無いとそもそも HDD が接続できません

USB キーボード:PC でも使えるらしいです。
USB マウス:これも PC でも使えるようです。特にハードウェアチェックなどはしていないようです。
VGA アダプタ:有名な話ですが、出力が Sync On Green です。UNIX 系 OS をいじっている方にはあまり問題とならないかもしれませんが、一般的にはあまり意識しないところなので対応モニタがないと厳しいかもしれません。
ただし、実は普通に NTSC(つまりはテレビ)に画面を表示することも出来ます。(後述)
BBN の CD-ROM:一度インストールをした後、HD を初期化するために必要です。上記に書いたとおり、初期のもの(あるいはβのことかもしれませんが)は、HDD ユーティリティディスク(実質は BBN の 1.0 か?)なるものがあるようで、あるいは必要ないケースもあるかもしれません。
Linux(1.0)インストールメディア:Version1.0 の場合は 2 枚組みの DVD-ROM です。β は DVD1 枚ですので、起動やインストールはβのほうが若干お手軽かもですね。

PS2 関連

PS2 専用 8MB メモリーカード 2 枚:上記のとおり、BBN を使用して HDD を初期化しますので、その際にメモリーカードが初期化されてしまいます。そのため 2 枚用意しています。
PS2 コントローラー:起動時に必要です。

その他

ネットワークケーブル(カテ 5)
VGA ケーブル(必要に応じて)
音声用同軸ケーブル:赤白。別途必要。今回は音声に関しては考慮していないので使用していません)
PC(Linux 稼動マシン):上記のとおり、コピー先として必要です。尚、同じサブネット上に無いとダメというか面倒なようです。

3.Linux とゲームデータの共存インストール

(注意)この手順は、http://achurch.org/ps2/linux-install.txt をかなりの部分参考にしています。
この実験については当方では方法を考案したわけではなく、実機での検証を目的としたものです。

.1.インストーラの起動
まずは Linux をインストールしないと始まりません。
そこでインストールメディアをマシンに投入しリセットボタンを押して、DVD-ROM から起動します。
このとき、PS2 コントローラーのボタンを以下のように押すことで、起動モードを変更できます。
VGA　　Select + L1
NTSC　　Select + R1
他にもありますが、日本国内で通常使うのは上の 2 つだけかと思います。これでその気になればテレビ画面を使用してインストールをすることも出来ます。少々文字が見づらいですが。。。
尚、起動画面表示種類を変更すると、次回以降もおなじ方法で起動しますので、TV→モニタ、あるいはその逆にしたい場合は、起動時に再度上記操作をする必要があるようです。

Linux1.0 の場合は、DVD2 枚組みになっており、起動は Vol1 から行います。
起動時にインストーラが起動するまでが、以下になります。
Disk1 で起動
Disk2 に入れ替え
Disk1 に戻す
これでインストーラが起動します。若干面倒ですね。この辺はβのほうがラクかもしれません。

2.インストールの開始
インストーラの起動後は、よくある昔の RedHat 系のテキストインストール画面です。ベースは Kondra MNU/Linux であるとの情報があるようですね。
最初は言語選択ですが、英語しか選べません。次にキーボード選択ですが、デフォルトで jp106 が選択されていますのでそのままにしておきましょう。
その後すぐに Disk2 に入れ替えが指示されます。基本的にパッケージ類はディスク 2 に入っているようです。
しばらく読み込むと、インストールタイプの選択に進みます。サーバか、ワークステーションか、カスタムの 3 種類が選べますが、ここでは動くの優先ですし、めんどいので「カスタム」を選択し、用意されているものは全部入れてしまいましょう。
3.パーティションの作成
次に、パーティション設定画面になりますが、ここですぐには次に進まず、
Alt+F2
を押して仮想コンソールを切り替えます。シェルが起動されていますので、ここで PS2 形式にあわせたパーティションを設定してしまいます。手順は以下になります。

a.「mknod /dev/hda b 3 0」で HDD 用デバイスファイルを作成する。
b.「fdisk -u /dev/hda」で、fdisk をセクタ単位モードで起動する。
c.パーティションを PS2 形式に合わせて作成する。(次ページ一覧表参照)

PS2 では、パーティションは以下の決まりがあります。
○128MB の倍数でなければならない。
○最初の8セクタはパーティションデバイスに含まれない。
○パーティションは複数のサブ・パーティションから構成されることが可能で、1つのサブ・パーティションのサイズは 128、256、512、1024MB のいずれかである
○従って、1GB を超えるパーティションは必ず複数のサブ・パーティションから構成される。
○複数のサブ・パーティションを利用する場合、各サブ・パーティションから最初の8セクタをパーティションデバイスから除く。
この条件に従って計算されたのが次ページの表になります。

サイズ	セクタ数	サイズ	セクタ数	サイズ	セクタ数
128MB	262136	5120MB	10485720	21504MB	44040024
256MB	524280	6144MB	12582864	22528MB	46137168
384MB	786416	7168MB	14680008	23552MB	48234312
512MB	1048568	8192MB	16777152	24576MB	50331456
640MB	1310704	9216MB	18874296	25600MB	52428600
768MB	1572848	10240MB	20971440	26624MB	54525744
896MB	1834984	11264MB	23068584	27648MB	56622888
1024MB	2097144	12288MB	25165728	28672MB	58720032
1280MB	2621424	13312MB	27262872	29696MB	60817176
1536MB	3145712	14336MB	29360016	30720MB	62914320
1792MB	3669992	15360MB	31457160	31744MB	65011464
2048MB	4194288	16384MB	33554304	32768MB	67108608
2560MB	5242856	17408MB	35651448	34816MB	71302896
3072MB	6291432	18432MB	37748592	36864MB	75497184
3584MB	7340000	19456MB	39845736	38016MB	77856464
4096MB	8388576	20480MB	41942880		

4096MB（4GB）パーティションの作成例。fdisk コマンド立ち上げ後の入力です。太字が入力した文字になります。

Command (m for help): **n**
Command action
e　extended
p　primary partition (1-4)
p
Partition number (1-4): **1**
First sector (63-78124094,default 63):　　←デフォルト値を使用するので空 Enter します。
Using default value 63 Last sector or +size or +sizeM or +sizeK (63-78124094, default 78124094): **+8388575**

最初の「n」で新規パーティションの作成を指定し、プライマリパーティションの 1 を作成します。
最初のセクタはデフォルトの数値をそのまま使用します。(63)
最後のセクタは、「+8388575」と、作成したいパーティションサイズをセクタ数で、相対値で作成します。
(相対値で作成するので、指定するセクタ数は上記表の数値から 1 を引いたものになります)
作成したパーティションを「p」コマンドで確認する場合、ブロック数はセクタ数の半分になっていることを確認して下さい。

```
   Device Boot      Start         End      Blocks   ID  System
/dev/hda1               63     8388638     4194288   83  Linux
```

上記であれば、
4194288x2=8388638-63+1
となっていれば問題ありません。
尚、確認時や、次項のパーティションテーブル書きこみ時にエラーが出ることがあるようですが、無視してかまいません。
当方では「w」を押すなりエラーが出ましたが、特にその後問題はありませんでした。
パーティションの作成が終わったら、「w」コマンドで、パーティションテーブルを HDD に書き込み、「Alt-F1」でインストーラに戻ります。

4.インストール用パーティション作成の完了
インストーラで「fdisk」を選択し、次の画面で「Done」を選択すると、パーティション一覧が表示されるので、「3」で作成したパーティションのマウントポイントを設定し(通常は「/」でしょう)、更に Swap パーティションを作成します。
尚、Swap パーティションはデータがコピーされるわけではないので、PS2 形式にこだわる必要はないようです。もちろん先に上記方法で作成しておくことも出来ます。

5. インストールの完了

「Done」を選択し、以降のネットワークの設定や、root、ユーザーの作成やパスワード設定等を行い、インストールを通常通り完了させます。

ちなみにかかる時間は、全部インストールして、この段階から大体 1 時間前後でしょうか。スペック差があるわけではないので、おそらくどなたがやっても同程度だと思います。

6.PS2 を再起動し、root でログイン

http://achurch.org/ps2/

から「initrd 起動用メモリカードファイルシステム」を適当な場所にダウンロードします。(環境に適切なものを一つ選択)。

当方は

http://achurch.org/ps2/ps2linux-boot-kanji-vga.tar.gz

を使用しましたが、通常は

http://achurch.org/ps2/ps2linux-boot-vga.tar.gz

で良いかと思います。

これをメモリーカード内に展開します。PS2Linux で上記 URL から直接ダウンロードしてもいいですし(wget は入ってたかな?)、折角もう一台 Linux マシンがありますから、そこにおいて ftp で get してもいいですし、方法はよしなに。

```
mount /mnt/mc00
rm -f /mnt/mc00/*
tar Cxzvf /mnt/mc00 /tmp/ps2linux-boot-vga.tar.gz
```

メモリカードをマウントし、中身を全削除して、先ほどダウンロードしたものを展開します。起動カーネルを消しているので、上記作業が終わる前に電源切ったりすると Linux が起動不可になるのでご注意。

7.PS2Linux を再起動

起動メニューに「Initrd」が増えていますので、これを選択すると、メモリカードからディスクレスでも動作する Linux が立ち上がります。立ち上がった Linux に、適当な名前を入力してログインします。必要に応じて ifconfig コマンドなどで IP アドレスを変更してください。デフォルトは 192.168.0.3/24 に設定されています。

8.Linux パーティションのデータの移動

用意した PC の Linux へデータを移動します。Linux では大体必要なコマンドはデフォルトで持っていると思いますが、nc(netcat)は入っていないかもしれません。各自でインストールしてください。当方は Debian を使用していたので apt で入れてしまいました。

で、以下のコマンドを順番に投入します。

```
PC 側> nc -lp12345 | gzip -1 > ps2-hda1.gz
PS2 側> dd bs=4M if=/dev/hda1 | nc 192.168.0.123 12345
```

192.168.0.123 は PC 側のアドレスです。各自の環境に合わせてください。

上記の要領で、Linux で使用するパーティションをすべて転送してください。当方は/パーティションと/swap パーティションのみでしたので、2 回行いました。当然ですが ps2-hda1.gz は別のファイル名で転送してください。

尚、転送が終わってもプログラムは終了しないので、dd の「xxx blocks out」メッセージが表示されたら、送信側の PS2 のプログラム実行を「Ctrl+C」で終了させて下さい。受信側を終了させるとデータ損失が発生するため、必ず送信側で終了させたほうが良いようです。

転送後のファイルサイズは大体 500MB 前後に収まるようですが、実際には余裕を持っておいたほうがよいかと思います。

8.HDD の再初期化

転送が終了したら、メモリーカードを入れ替えて、BBN の CD を挿入して再起動しましょう。BBN のインストール処理が始まり、HDD の初期化と BBN のインストールが行われます。

初期のものには HDD 初期化ツールがついていたようですが、当方の環境には BBN しかないため、上記の方法をとっています。HDD 初期化ツールがあるならそちらを使ったほうが早いのかもしれません。

9.Linux 環境の書き戻し
HDD の初期化がすんだら、再度 DVD-ROM とメモリカードを入れ替えて、先ほどの 項番 7 と同様に、「「Initrd」から起動します。
起動した Linux で、ps2fdisk コマンドで Linux 用のパーティションを作成します。サイズは 3 番で作成したのと同じサイズ・パーティション番号としてください。
尚、ps2fdisk にはこまかいオプションはありません。ので、サイズ決定の際に、作成したパーティションと同じサイズを MB 単位で入力します。ps2fdisk 終了時、新しいパーティションテーブルが正常に読み込まれたことを確認します。正常に読み込まれなかった時は 14 番と同じ要領で再起動を行ってください。

10.Linux パーティションのデータを PC から PS2 に転送
以下の順でコマンドを投入します。
PS2 側> nc -lp12345 | reblock > /dev/hda1
PC 側> (zcat ps2-hda1.gz; echo >&2 done) | nc 192.168.0.3 12345
ここでは PS2Linux の IP アドレスが、ツールデフォルトの IP アドレスを指定していますが、項番 7 に記載のとおり、変更することも出来ます。
PC 側のコマンドに「echo >&2 dnoe」を入れてあるので、転送が終了すると「done」と表示されます。最初の転送時と同様、プログラムは自動では終了されませんので。送信側の PC のプログラム実行を「Ctrl+C」で終了させて下さい。
また、当然ですが、必要なパーティションすべて書き戻しましょう。
書き戻しが完了したら、再び Linux を再起動し、今度は通常の起動を行います。正常に立ち上がってくれば、とりあえず Linux の導入には成功です。

11.BB ナビゲーター
現時点では、Linux が立ち上がるのみで、共存できているとはいえません。BBN が起動できないからです。これは Linux の場合と同様、BBN を起動するためのメモリーカードが必要だからです。
再度メモリーカードを交換し、BBN 起動用 CD-ROM を入れて再起動します。
BBN メニューが立ち上がってきますので、「起動用メモリーカードの作成」を選択し、BBN 起動用のメモリーカードを作成します。
*尚、当方では Linux インストール後に行ったというだけで、もしかすると項番 8 で、再度 BBN の CD から起動することで上記作業は可能かもしれません。とにかくも、どの段階でもよいですので、BB ナビゲータの起動メニューから起動用のメモリーカードを作成することで、起動が可能になります。

12.以上がすべて終了すると、とりあえずの共存インストールは終了です。これで、BBN メモリーカードを使用すると BBN が立ち上がり、Linux-DVD+Linux 起動用メモリーカードを使用すると、Linux が立ち上がってきます。

4.Linux を DVD-ROM 無しで起動できる状態にする

さて、前項までで、とりあえずの共存インストールは終了しましたが、このままではそれぞれの起動時にメモリーカードや DVD を交換する必要があり、非常に面倒です。
さりとて、DVD-ROM はいわゆるブートローダーのようなもので、これ無しでは起動が出来ません。では、どうすればよいか？
これまでの記事の中で書いたとおり、「Playstation BB ナビゲーター」の実態は Linux です。そもそも PS2Linux 自体が BB ナビゲーターの開発の副産物ではないかとさえ言われているくらいです。
で、実質 Linux である BB ナビゲータは、DVD 無しでも PS2 を起動できます。これを利用して、BB ナビゲータから PS2Linux を起動してしまおうというわけです。

1.カーネルのアップグレード

2.4.17 のカーネルを再構築し、カーネルをアップデートします。その際にパッチを適用し、BBN のパーティションもマウントできるようにします。
何故このようなことをするのかというと、DVD-ROM 無しで起動させるには上記のとおり BBN から起動する必要がありますが、その際に BBN の起動スクリプトをいじる必要があるためです。

まず、ソースコードとカーネルヘッダーを持ってきます。
http://www.sony.net/Products/Linux/Download/PlayStation_BB_Navigator/kernel-headers-2.4.17_ps2-26.mipsel.rpm
http://www.sony.net/Products/Linux/Download/PlayStation_BB_Navigator/kernel-source-2.4.17_ps2-26.mipsel.rpm
ちなみに wget で直接持ってこようとしても NG でした。
当方は PS2Linux にログインして/etc/inetd.conf を書き換えて Telnet と ftp を有効にして、リモートから接続して実施しました。ファイルも最初にダウンロードしておき、ftp で転送しています。方法はよしなに。先ほどのインストールで用意した別マシン経由でも良いかもしれません。

rpm から強制的にファイルを取り出してみますが、おそらく rpm でそのままインストールしても多分大丈夫だと思います。

```
# cd /var/tmp
# rpm2cpio kernel-headers-2.4.17_ps2-22.mipsel.rpm | cpio -i --make-directories
# rpm2cpio kernel-source-2.4.17_ps2-22.mipsel.rpm | cpio -i --make-directories
# cd usr/src
# mv linux-2.4.17_ps2 /usr/src
# cd /usr/src
# mv linux linux.org                    →以前のカーネルソースへのシンボリックリンクを待避します。
# ln -s linux-2.4.17_ps2 linux          →2.4.17_ps2 へのシンボリックリンクを作成します。
```

HDD フォーマット認識が BB Navi 仕様になっているので、PS2Linux 仕様を読み取れるようにパッチをあてます。
http://hp.vector.co.jp/authors/VA008536/ps2linux/bblinux3.diff.gz
http://hp.vector.co.jp/authors/VA008536/ps2linux/bblinux4.diff.gz
*どちらのほうがいいかはわかりません。当方は「怪しいパッチ」とかなんいいける bblinux4 でテストしています。これは BB Navigator の "_linux.X" というIDのパーティションを認識できるようになるらしいですが、危険がいっぱいなので自己責任でお願いします。とのことです。

```
# cd /usr/src/linux
# patch -p0 < bblinux4.diff
# cd scripts
# rm mkdep split-include                →実行できないので
# cd ..
```

ここから先は普通のカーネル再構築になります。

```
# cd /usr/src/linux
# cp -p arch/mips/configs/defconfig-ps2 .config
```

大抵の環境では ext2fs を組み込みにする必要が有ります　→.config の CONFIG_EXT2_FS=y に変更します。
また、カーネルオプションが
root=/dev/hda1 crtmode=ntsc graphics CONSOLE=/dev/null
となっているので、環境に合うように変更をしたほうが良いようです。
当方は同様の環境なのですが、とりあえず
crtmode=ntsc
にだけ変更しています。
vi /usr/src/linux/arch/mips/ps2/sbios/init.c （カーネル 2.2.*の場合）
vi /usr/src/linux/arch/mips/ps2/prom.c　　（カーネル 2.4.17 の場合）

```
    /* get command line parameters */
        if (ps2_bootinfo->opt_string != NULL) {
            int i;
            for (i = 0; i < CL_SIZE - 1 && ps2_bootinfo->opt_string[i]; i++)
                arcs_cmdline[i] = ps2_bootinfo->opt_string[i];
            arcs_cmdline[i] = '¥0';
        }
        strcpy(arcs_cmdline, "root=/dev/hda1 crtmode=ntsc");  ←この一行を加える。
```

make oldconfig　または make menuconfig　　→オプションで必要そうなものがあれば好きに設定してください。ただし上記に記載した必須項目を変更しないように。
make dep
make clean
make

make modules
make modules_install
mount /mnt/mc00
gzip -c9 vmlinux > /mnt/mc00/vmlinux-2.4.17.gz　→圧縮したカーネルを作成（未圧縮でコピーでも可）
vi /mnt/mc00/p2lboot.cnf（2行目の）vmlinux-2.4.17.gz のエントリを追加。無論正確に上記のカーネルを示していれば最初の" Linux 2.4.17"や、" Linux-2.4.17 on Memory Card(PS2)"の書き方は任意でかまいません。また、下記は普通の Boot 用のメモリーカードの例ですが、ファイルのサイズさえきちんと調整できれば、最初の実験で使用した、initrd ファイルシステムのカードにも入れられます。
当方は initrd ファイルシステム用のカーネルも圧縮し、p2lboot.conf の通常 Boot 他の記述も書き換えて、使用しました。

```
"Linux on MC" vmlinux "" 203 /dev/hda1 "" Linux on Memory Card(PS2)
"Linux 2.4.17" vmlinux-2.4.17.gz "" 203 /dev/hda1 "" Linux-2.4.17 on Memory Card(PS2)
"Single" vmlinux "" 203 /dev/hda1 single Single User Mode
"Emergency" vmlinux "" 203 /dev/hda1 emergency Emergency Mode
```

umount /mnt/mc00

これでカーネルがアップデートされたはずです。起動時に「Linux-2.4.17 on Memory Card」を選択してみましょう。
（無論、任意に書き換えていればそれを選択）
新しいカーネルで起動し、Montavista Linux のロゴが出るようになるはずです。

2.akload の make

さて、カーネルをアップデートしたことで、reiser fs をマウントできるようになりました。これで、BBN から起動するための細工を、BBN 側パーティションに施すことが出来ます。

まず、BBN から ps2linux を起動するために埋め込むためのブートローダー、akload を make します。

http://hp.vector.co.jp/authors/VA008536/ps2linux/akmem_ps2.tar.gz

上記の URL は、PS2Linux 用の akload です。こちらをダウンロードしてインストールします。

先ほどのカーネルについてもそうですが、PS2Linux 環境に持ってくる方法は任意の方法で。Wget でも ftp で PS2 に Put してもなんでもかまわないと思います。

Akload のインストール時には、以下のようになっている必要がありますが、カーネルをアップデートした状態であればこうなっているはずです。

```
# cd /usr/src
# ls -l
lrwxrwxrwx    1 root     root           16 xxx xx xx:xx linux -> linux-2.4.17_ps2
drwxr-xr-x   15 root     root         4096 xxx xx xx:xx linux-2.4.17_ps2
```

/usr/src/linux が 2.2.19 または 2.2.21, 2.4.17 のカーネルを指している必要があります。上記カーネルでない場合は読み替えてください。尚、BBN からの起動を行うには 2.2.*以上である必要があります。(reiserfs をマウントできる必要があるからです)

上記の状態であれば akload を make します。

```
# tar zxf akmem_ps2.tar.gz
# cd akmem_ps2
# make
# make mknod
# make install （/sbin/akload にコピーするだけですので手動でも可）
```

3.BBN からの起動設定

次に BBN のメモリカードをマウントして akload をコピーします。

```
# mount -t reiserfs /dev/hda11 /mnt/bbn
# cp -p /sbin/akload /mnt/bbn/sbin
```

上記は何故 /hda11 なのかについても詳しく書きたいところですが、とりあえずは簡単に。

PS2 の BBN のファイルシステムである「_linux.X」のパーティションは 「/dev/hda11～」に割り当てられます。

基本的なルートパーティションは「_linux.1」にありますので、「/dev/hda11」をマウントすることで、BBN のルートパーティションにアクセスできます。

次に Boot させたいカーネルを BBN のパーティションにコピーします。

```
# cp -p /usr/src/linux/vmlinux /mnt/bbn/boot/ps2linux-2.4.17
```

BBN の起動スクリプトに起動時に別カーネルを起動するスクリプトを記述します。

おそらくいろいろな方法があると思いますが、とりあえず当方では以下の方法で行っています。

「# Boot PS2 Linux～」からの 3 行を挿入しています。

```
#vi /mnt/bbn/etc/rc.d/rc.sysinit
# copy bn_asap to /sbin
        if [ -f /opt0/bn/bn_asap -a ! -f /sbin/bn_asap ]; then
                /opt0/bn/bn_asap > /dev/null 2>&1
                /bin/cp -p /opt0/bn/bn_asap /sbin
                /bin/chmod u+s /sbin/bn_asap
        fi

        >/etc/mtab
        /bin/mount -a -n -t nonfs,smbfs,ncpfs,proc

        # Boot PS2 Linux if Controller bunnton is pressed
        BUTTON=`cat /proc/ps2pad | awk '$1==0 { print $5; }'`
        [ "$BUTTON" != "" -a "$BUTTON" != "FFFF" ] && /sbin/akload -r /boot/ ps2linux-2.4.17  ←この 3 行を追記
```

```
/bin/rm -f /var/run/bn.pid /var/run/runlevel.dir /var/run/setcrtmode
/var/run/klog.pid /var/run/syslogd.pid
/bin/rm -f /var/lock/console/* /var/lock/samba/* /var/lock/subsys/*
/bin/rm -f /var/run/netreport/*
/bin/rm -f /tmp/.X*-lock
/bin/rm -f /tmp/[0-9a-f][0-9a-f][0-9a-f][0-9a-f][0-9a-f][0-9a-f][0-9a-f][0-9a-f]

# Clean up utmp/wtmp
```

上記のスクリプトにより、起動時にコントローラのボタンを押していると、PS2Linux が起動し、何もしないでいると BBN が起動してくる環境となります。
これで、BBN から起動し、かつ 2.4.17 カーネルになり、かつ共存された環境で Linux が使用できるようになりました。

なお、当方はすべてコンソールで作業を行っていますが、もし GUI を使うなら、USB マウス周りの変更が必要(デバイス番号が変わってるらしい)であるとの情報があります。

5.終わりに。

とりあえず以上で、今回 PS2Linux で行う実験は終了とします。ポイントは、やはり DVD-ROM 無しで Linux が立ち上げられるという点に尽きるでしょう。
ただし、当然といえば当然ですが、基本的に中身を見られることを前提としていないパーティションを勝手に見るわけですから、ファイルシステムが何らかのきっかけで破壊されてもまったくおかしくありません。あくまで試してみる際には自己責任での作業をお願いします。

また、実は今回、カーネルアップデート後の状態で若干の異常がありました。
当方の実験を進める上では問題なかったので、あえて文中には書いていませんが、もしかすると上記のとおりやっても結構クリティカルな間違いをしている可能性もあるのでご注意ください。

若干の異常とは、以下の 2 点です。
1. SWAP をマウントできていない
2. 起動時に devpts の mount 異常が起きている

カーネル再構築時のオプションの設定で微妙な間違いをしてしまっている可能性が高いと見ています。どちらも失敗していても起動、動作にさほどの問題はありませんが、SWAP をマウントできていないので、あまりにメモリを食うような処理はうまくいかない／時間がかかる可能性がありますし、devpts をマウントできていないので、GUI を使用した際に端末エミュレータが動作しません。
この辺はまた別途、カーネル再構築の際のオプションについて調査してみたいと思いますが、もしお試しになる場合は上記の点には気をつけてください。もしくは詳しい方、お教えくださると感謝の極みです。

ともあれ、念願かなって PS2Linux をいじることが出来ました。今後も継続して実験を行っていきたいと思っています。
とりあえずは正常に Boot するようにカーネルを作り直し、その上で GUI 起動を試みる所存です。
最終的には GUI は
http://lainos.sourceforge.net/
をインストールして使用できれば面白いなぁと思っていますが、これはこれで敷居が高そうではあります(汗
また、現状提供されている以上のカーネルなども、海外のフォーラムなども参考に見たいと思っています。まあそういっているうちに今度は PS3 に浮気しそうではありますが。。。

＜了＞

OS としての VMWare ESX Server

りろ@涅槃

1 まえがき

仮想マシンソリューションが熱いらしい。仮想化技術はハイエンドサーバ分野では昔から実装されているが、いわゆるエンタープライズ環境だけではなく、我らマイナーOS 愛好家も仮想マシンの普及によって相当の恩恵を受けている。ついぞ3年前ならば、素性の怪しい OS を試すのに、空いている実機を探し、無ければ中古 PC 屋でわざわざ物色してくるか、自作するか、リスクを承知でマルチブート環境で運用するのが普通であった。筆者のサイトで公開しているコンテンツ「マルチブートの庭」は、そもそもマイナーOS を試すのにマルチブート環境を構築していたのが発端となって、いつしか手段と目的が逆転し、様々のマルチブート実験を試みた結果である。

現在では、仮想マシンソリューションベンダーの雄 VMWare が VMWare Player と VMWare Server を無償提供するという画期的な状況となり、マイナーOS の一般向けリリースは、LiveCD イメージや VMWare イメージで配布されることが多くなった。仮にバイナリでしか配布されなくても、インストール媒体が提供されれば、VMWare Server を利用して仮想マシンを構築できるようになった。Solaris10 などは VMWare 用のグラフィックドライバを最初から搭載しているほどだ。

現在、我らマルチ OS/マイナーOS 愛好家は、わざわざテスト用の実機をいくつも用意する必要はなくなり、十分な主記憶と補助記憶装置を積んだマシン1基を用意しておけばほとんどのテスト～実運用まで行えるようになった。我が家などは、かつて9台のマシンが狭い部屋を占領し、足の踏み場もないほどであったが、現在は人間らしい生活のできる広いマシンルームで実験を行っている。この空いたスペースに中古の UNIX サーバを置きたい誘惑にしょっちゅう駆られるが、家人から「気の流れが悪くなる」と戒められ、敗退を重ねている（滂沱の涙）

さて、今回のテーマは、VMWare がかつての主力製品のひとつであった VMWare GSX Server を無償提供してまで売り込みをかけた VMWare ESX Server である。

当然のことであるが、VMWare は我らマイナーOS 愛好家のために英断をふるったのではなく、オープンソースの仮想マシンソフト Xen の猛追や、次期 Windows Server での仮想化環境標準搭載などの煽りを受けて、主力商品であるエンタープライズ向け仮想化ソリューション VMWare Infrastracture と、そのスイート群の中核である VMWare ESX Server の売り込みを強化するためである。

次項で述べるが、VMWare ESX Server はホスト OS を必要としないと思われているが、それもそのはず、それ自体が特殊な LinuxOS だからである。

2 VMWAre (GSX) Server と VMWAre ESX Server はどう違うか

ごく簡単に言ってしまえば、VMWare (GSX) Server は、既存のホスト OS の上にアプリケーションとしてインストールされて仮想マシンレイヤーを構築し、その上でゲスト OS を動かすものである。 CPU は2WAY の SMP として仮想化されるが、実運用に供されている重いホスト OS 上でゲスト OS が動作するものである以上、ネイティブ・インストール OS に比べてパフォーマンスの低下は避けられない。なので、1 台の物理サーバ上でホストを本番環境、テスト環境、開発環境をそれぞれゲスト OS として立てるなどの用途で用いられるのが基本である。

これに対して、VMWare ESX Server はデータセンタクラスの環境で仮想化環境を提供するもので、ホスト OS を必要とせずに仮想マシンレイヤーを構築する。ただしここで誤解が生まれるのだが、VMWare ESX Server はホスト OS を必要としないのではなく、正確にはそれ自体が特殊な OS なのである。Linux カーネル上に仮想マシンレイヤーとモニタリング機能を実装している。最大論理 CPU32 個という強大な仮想化機能と、仮想マシン実体を SAN などの高速ストレージに置くなど、データセンタクラスの環境で力を発揮する製品である。

なお、VMWare (GSX) Server は現在無償提供されているが、これは ESX Server への移行を支援するためである。

3 VMWare ESX Server のインストール

怖がることはない。Linux のインストールとほとんど同じである。CD-ROM からブートして、指示に従っていけばインストールは終了する。

以下、順次説明する

○CD-ROM からブート

Linux とほぼ同じ画面を拝むことが出来るが、ブートオプションとしていくつか用意されている

【TEXT】テキストモードでのインストールを可能にする。ESX Server 自体に GUI は必要としないのだが、インストーラだけは GUI が用意されている
【DRIVERDISK】特殊なデバイスのためのドライバ適用
【SANBOOT】SAN ストレージからのブートを可能にする場合に選択

○ライセンス確認とシリアル入力

商用製品お約束のライセンス確認とシリアル入力画面がある。指示通りに入力

○パーティショニング

デフォルト通りのパーティショニングを試みたところ、ほとんどの領域は、拡張パーティションに割り当てられた。ここが仮想マシンファイルの置き場所となるはずである。実運用ではこのパーティションには RAIDO+1 または RAID5 または 6 が割り当てられるであろう。SAN の冗長性確保がどうなっているのかわからないが、いずれにしても基本は同じである。

○インストール開始

特にすることはないので、鼻くそほじって眺めていてかまわない。比較的簡単にインストールは終了する

○パスワードとアカウント

root のパスワードを設定する。必要であれば一般ユーザーアカウントも作成できる

○インストール終了とログイン

VMWare のコンソールは管理専用端末からブラウザで行う。http:// IPADRESS/ で GUI ログイン画面が出るが、それは後にして、Linux としての VMWare ESX Server を探索してみよう。ALT+F2 を押すと、通常の CUI ログイン画面が出るので ROOT でログインする。まずは起動している主なプロセスを見てみた。httpd xinetd syslogd のほか、vmware-server もデーモンとして起動している。さらにジャーナリングを行うデーモンなども常時起動している。スーパーデーモン xinetd が管理しているのは、chargen echo time などのスモールデーモンに加えて、wu-ftpd と VMWare の WEB ログイン認証デーモンがあるようだ。

当然ながら余計なプロセスは立っていないので、軽量のように見えるのだが、この仮想マシンモニタ(Linux)用のリザーブメモリーが最低 192M なので、決して軽量ではない。

○/直下の各ディレクトリ拝見

【bin】一般的なコマンドが置かれている

【boot】ブート関連ファイル格納ディレクトリである

【dev】デバイスファイルが置かれているが、たまげたのはハードディスク関連ファイルの多さである hda1～hdt9 まで単純計算で 200 基以上の物理ハードディスクが接続できることになる。データセンター向け製品ということだからある意味当然なのだろう。

【etc】UNIX の世界ではこのディレクトリは魔境なので、あまり深く掘るのは憚れるが、とりあえず気がついたことをいくつか

○なぜか SAMBA のコンフィグファイルがある。VMWare 自身が使うのか、単なる Linux の痕跡なのかは不明

○WEBMIN がインストールされているようだ。WEBMIN は高性能管理ツールと称してハイエンドサーバ OS にも実装されているが、VMWare ESX Server も例外ではない模様

○VMWare のコンフィグファイルもここに置かれている(/etc/vmware)

【home】一般ユーザーのホームディレクトリである。デフォルトの一般ユーザーとして vmware がお住まいになっておられるようだ。なお、/home/vmware の下に運用する仮想マシンの実体が置かれるので、デフォルトの場合には、このディレクトリに高性能ストレージがマウントされるはずである

【lib】ライブラリ用ディレクトリである

【mnt】デフォルトでは CD-ROM と FLOPPY がある /etc/fstab にもマウントが記述されている

【proc】プロセスファイル用ディレクトリである

【sbin】一般的な管理者コマンドが置かれている

【usr】/usr/bin /usr/sbin には、VMWare 関連のコマンドが置かれているほかは、空っぽのサブディレクトリが多い。一応 /usr/X11R6 があるので、その気になれば GUI 付き Linux としても動作するのだと思われる。

【opt】空っぽである

【root】root のホームディレクトリである

【tmp】テンポラリディレクトリである

【var】基本に忠実にログ格納庫であるが、pegasus のログ格納庫があった。Openpegasus は、RedHat Enterprise などに実装されているリソース管理ソリューションである。VMWare で実際に使われているのかどうかは不明だが、VMWare ESX Server が想定する環境は、極めて複雑なリソース管理が要求されるはずなので、内部で使っているのかもしれない。

4 VMWare Infrastructure による仮想マシンソリューション

詳細については、拙サイト「未来のユートピア的・ノスタルジー的遠方」において実験結果を公開する予定であるので、興味を持っていただける方がいらしたら、ご高覧賜りたいと思う。

http://www.sternklang.net/chubo/imin/

VMWare ESX Server が提供する仮想マシンレイヤーでの OS インストールは、これからじっくり時間をかけて調査するので、本稿では割愛させていただく(というか間に合わなかったので陳謝)

VMWare のサイトでアナウンスされている VMWare Infrastructure の導入について簡単に説明しておきたい。

○VMware P2V Assistant	物理マシンの構成を VMWare 仮想マシン環境に変換
○VMware VMFS	SAN ストレージ上に複数の ESX Server プラットフォームを提供
○VMware VMotion	仮想マシンに障害が起きたときにダウンタイムなしで待機系の仮想マシンに移行
○VMware Virtual SMP	複数仮想マシンの CPU 資源最適化静的ロードバランサ
○VMware HA	物理サーバダウン時のフェイルオーバ
○VMware DRS	ダイナミック・ロードバランサ

これらを全て実装したデータセンターの管理は複雑を極めるので、VMWare 認定技術者の常駐が不可欠であろう。汎用機からクライアントサーバへそして仮想化ソリューションへ。一度汎用機から離れて独立したサーバ群の集約化が進んでいるように思えるが、どうやっても金はかかるのである。

VMWare ESX ServerあるいはVMWare Infrastructureの試用版は容易に入手できるが、本名と会社名を明かした上で、VMWare 社からの確認電話を受ける必要がある。したがって、電話で導入予定などを尋ねられて「単なる遊びです」などと答えた日には、冷やかしと認定されるので注意が必要であるというトホホな警告を書いておきたい。

求む！マイナーOS 情報

立神梢一

さて、当方ではマイナーな OS についての研究、調査、コレクションをしているわけですが、いくつかの OS については充分に情報がなかったり、入手が困難だったり、ハード面で検証が厳しかったりといったことがあります。
本稿ではそうした情報をまとめて、各種研究の礎にしたいと考えて、記したものです。皆様からの情報をお待ちしております。
また、情報と言っても、専門的な事に止まらず、いつ頃、どこで、どういったシーンで使われていた、といった事も大歓迎いたします。

1 情報が不完全なもの

まずは、当方にて一部入手するなどしていますが、情報がいろいろな理由で不完全な物についてです。

1-1.THEOS

イギリスの THEOS 社が開発している OS のようです。
会社も存続していますし、WEB 上に情報は有りますが、マルチバイト圏はまるで眼中に無いようで、日本語リソースは殆んどありません。
昔は代理店が多少取り扱いもしていたようですが、今はないみたいですね。

で、バイナリは WEB 上にあるのですが、どうもハードウェアドングルが必要なようで、インストール出来ません。価格も今ひとつ見えないのですが、安くはないでしょうし。

雰囲気としては NETWARE 的なサーバーOS のようですね。
使っていた、所持している、等々情報をお持ちでしたら是非お寄せ下さい。

1-2.VINES

NETWARE との覇権争いに破れ、1998 年になくなった、BANYAN 社の OS です。ディレクトリサービスを最初に実装し、価格も抑えた OS だったようですが、マーケティングのまずさで負けてしまったようです。
当方はパッチ CD とメールサービスらしきソフトウェアのみ所持しております。OS 本体は所持していません。
また、この OS もハードウェアドングルが必要との情報もあり、上記 1 と似たような系列なのかな？と思ったり。
ただ VINES は基本的には UNIX だとのことなので、全然違うのかもしれませんが。
1998 年までは日本語版もあったわけですし、ご存じの方、是非情報をお寄せ下さい。

1-3.DASCOM AD

1996 年頃、DASCOM 社が出していた UNIX です。日本ではセコム情報システムが代理店をしていたようです。

2000 年問題非対応です。ほかにもいろいろ芳ばしいですがまあ古いので仕方ない。

まず、DASCOM 社はすでに IBM に買収されております。またこの OS はそもそも WEB STARTER キットという WEB サーバ用パッケージとして発売されたようです。
DCE という暗号化関連テクノロジの実装パッケージのようです。IBM が買収したのは、このへんのこと込みだったのかな？
一応インストールは実験済みですが、使用実績など、より詳細な情報があればうれしいです。

1-4.SONY NEWS

SONY 製 UNIX です。先日 OS 本体は入手しましたがハードがほぼ絶望的な状態です。
万が一、所持していて譲渡していただける方←居ないか(笑)、所持していて使用している方、使用していた方の情報をお待ちしております。

1-5.EWS4800
OS は UX/4800 になるかと思います。
NEC 製の UNIX です。ハードウェアは意外とヤフオク等で目にしますが、OS はまったく見たことがないですね。
一時、ハードは所持していましたが、いろいろな都合上、手放してしまいました。残念です。
OS を所持している方、使用していた方の情報お待ちしております。
（今回、りろ氏からもすこしお寄せいただきました）

1-6.その他 OLD UNIX
十把ひとからげで申し訳ないですが、その他の UNIX です。
いわゆる UMIX 戦争以前に存在したものなど使用していた方の情報お待ちしております。

2 現行 OS であっても、主にハードウェア上の理由で検証、実験が不十分なもの。

2-1.MORPHOS
PPC 用の AMIGA 後継 OS、いわゆるペガサスシステムのマザーボード、アーキテクチャ用の OS です。
ペガサスシステムを購入すればバンドルされているらしいのですが、海外から直販でないと若干高くつくなどもあり、踏み切れておりません。冬に買っちまおうかなぁ…
一応、マザーボードなど取り扱っている国内ベンダもあるようですが。

2-2 RISC-OS
Acorn というアーキテクチャ用の OS です。組み込み系の CPU、ARM の会社という認識であってるかしらん。
これも、イギリスからの直販でないと購入出来ず、ハードウェアにバンドルはされているみたいですが、敷居は MORPHOS より高そうです。
以前日本の、RISC OS を扱ったサイトの方から情報を伺ったことがありますが、かなり大変なので、良く考えた方がいいと諭されたことがあります…
確か日本の方が作った画像ビューワがあったような…

2-3 組み込み系列のもの
非常に抽象的な書き方になってしまいますが、組み込み系で、ARM などの CPU アーキテクチャについてや、組み込み系 OS を PC を使用して起動した後に、テスト等を行うネタなどがある方の情報をお待ちしております。AzKiRTOS などがこれにあたるでしょうか。
当方があまり組み込み系、プログラミングそのものについては知識が無く、力をおかりしたいしだいです。

3 実験協力者、原稿執筆者募集

独自に原稿を書いて頂ける方や、当方の実験の手助けをして頂ける方を募集しております。やはり一人では手が回りませんし、また同系統の物でも、違う切口からの原稿等も歓迎致します。
またさらに、当方のネタについての検証や、（例えば、ビルド、起動ログやスクリーンショットの採取、使用コンパイラの情報を提供して頂けるなど）独自に得た情報等の出どころや経緯を断片的にでも頂けるなども大歓迎いたします。

まあ現状では相応のお礼は出来ないのですが…完成した本を差し上げる程度ですが、それでも何か情報を頂ける方をお待ちしております。

また、ココに記載がないマイナーOS や、あるいはマイナーOS の周辺記事についても歓迎いたします。
例えば、マルチ OS 環境用のブートローダについてや、GRUB による各種実験、ファイルシステムについてや階層構造など OS のアーキテクチャによる差異の研究等も、お待ちしております。
OS としての研究に加えて、、異種アーキテクチャについての情報もお待ちしております。すでに書いたいくつかの OS は、異種アーキテクチャ専用 OS ですし、BEBOX や NeXT などについても情報をお待ちしております。

以上が現在のマイナーOS 研究、進捗情報と募集事項となります

4 これからのマイナーOS 研究スケジュール

研究予定の内容

研究対象としては、大雑把に実験分類を行う
1
(必要なら)ビルド、インストール、起動までを主に確認する。+起動ログもしくはスクリーンショットを記事に添付

2 インストール、起動までを第一フェーズ、その後、第二フェーズとして、OS 上にて各種設定、環境構築までをある程度行う。

3 1とほぼ内容は同じだが、1FDOS など最初の段階にあまり手間のかからないもので、系統が近いもの等を複数まとめて記事にする。
いわゆるスクリーンショットコレクション的な内容

4 複数の OS の機能、ファイルシステム、ベンチマークテストなどによる能力比較、機能比較、等の研究。かなり高度な内容が要求されると思われる。

5 実際には所持していない、あるいは実験出来ていないものに関する記事。

具体的な OS 名と実験したい内容

4-1 AMOEBA
スタンドアロンインストール、X 導入など。
分散 OS なので一台で出来ることは限られていそうですが、実稼働をあまり聞かないので。仮想マシン環境でもいけるのか？どこかの大学では実験的な環境として構築されたりはしているようですが。
環境構築を目指せるか。

4-2 PLAN9
ZAEMON 氏の情報とかもらえないかなぁ。
インストールはしたことあり
これも、環境構築を目指せるかどうか。モノや資料は十分にあるので、あとは時間と根気ですかね。

4-3 BE、QNX、超漢字
3 つまとめてなのは、OS としての完成度は比較的上だから。
環境もだが、ポートとか比較とかも？前回、QNX についてはとりあえずひととおり試してはいますが、まだまだ出来ることは多そうです。

4-4 AROS、EOS、REACTOS
一応それなりの完成度があり、かつ開発環境がありそう？ということで。SYLLABLE とかもか？

4-5 VSTa、ForthOS
なんか敷居高そうだから。
インストールだけでも苦労しそう…
これ以外にも、JAVAOS とかもいろいろあり、ある程度環境が整って居るようならそれなりの実験をしたいですね。

また、並行してマイナーOS リストも充実させて行きたい所存です。
マイナーOS リストは紙媒体だとやりにくい面もあり、本誌には掲載しておりませんが、一応徐々に Web 上で拡充させております。情報をいただける方是非！
http://fnow.org/
Top からデータベース→マイナーOS と辿ると見られますです。

執筆者コメント

立神梢一

なんとか 2 号の発行にこぎつけることが出来ました。
元は昨年冬に発行予定だったのですが、コミケに参加できず 1 年越しの発行となってしまいました。
今回は急遽入手した PS2 と、長年の課題？だった ChorusOS の実験を行うことができ、それなりには満足していますが、もっとヘンテコな OS はまだまだたくさんあります。これに飽き足らずどんどんとヘンな実験をしていきたいと思っています。

りろ@涅槃

マイナーOS 本第2巻刊行　慶賀に存じます。
私の最近の関心事は仮想マシン運用に絞られていますので、第三巻刊行時には、複雑きわまる仮想マシン運用の話が書けたらいいなと思っています。

あとがきにかえて

立神梢一

あとがきとかそのほか
1 号に引き続き執筆していただいたりろ@涅槃氏、本当にありがとうございました。おかげさまで今回もそれなりのページ数が確保できました。
また、当方に PS2Linux をプレゼントしてくれた相方、本当にありがとうです。ついゲームしちゃうんだけどね。
今回は昨年冬に出すはずだったので、スケジュールとしてはラクでした。とはいえコミケでのマイナーOS 本以外にも、創作方面やアーケードゲームにも手を出しているのでなかなか暇はありません。。。
次号の OS ネタについては、すでに原稿の一部として書いていますので、それ以外のことを。。。
とりあえずなんといってもマイナーOS データベース。すこしづつ追記はしていますが、まだまだ追いついていない状態です。
まあそんなこと言ってるうちに OS 自体がなくなっているケースもままありますが。
そして文字ばっかりの本になっているので、もうすこし見栄えを良くしたいという気もしなくはないです。とりあえずは、スクリーンショットの掲載を中心に考えて生きたいと思っています。

当方の Web サイト、「Far Northern Other World」にて、マイナーOS データベースを中心に、今後もマイナーOS についての研究を続けていきたいと思います。よろしければお立ち寄り下さい。
http://fnow.org/
本冊子も、継続して発行していきたいと思います。
また、上記マイナーOS データベースへの情報提供、本冊子への原稿執筆、実験協力などをしていただける方を随時募集しております。
現状では、協力者への本冊子を差し上げる程度しかお礼が出来ませんが、ご参加をお待ちしております。

マイナーOS コミュニティ活動について
Mixi 等の SNS や、Google Groups において、「マイナーOS マニアックス」コミュニティを作成しております
ので、ご興味をお持ちの方は、どうぞご参加下さい。

Google Groups	http://groups.google.com/group/osmaniax
Mixi	http://mixi.jp/view_community.pl?id=550392
Otaba	http://otaba.jp/page.php?p=c_home&target_c_commu_id=2373
Filn	http://www.filn.jp/page.php?p=c_home&target_c_commu_id=3726
Livedoor フレパ	http://www.frepa.livedoor.com/community/index?community_id=7296

まあ正直なところ現状では Mixi くらいしかまともに活動はしていません。GoolgeGroups はスパムも多いし閉鎖予定ですが、一応まだ残してあります。
ですが、原稿募集、イベント参加通知等はとりあえずすべての SNS のコミュニティにて行っています。
今後は当方の Web サイトに集約していきたい考えです。

読了ありがとうございました。まだまだヘンな OS 実験を毎日行って次号に備えております。また Version.3 でお会いしましょう。
それではまた。

「Operating System Maniacs」 Version.2.0 奥付

発行日 2007 年 8 月 17 日(コミックマーケット 72) 東カ 48b
発行 Far Northern Other World ー極北別世界ー
http://fnow.org/
発行者 佐藤誠之
印刷 ねこのしっぽ 様
連絡先 141-0033
東京都品川区西品川 1-26-12
佐藤誠之 (as 立神梢一)

「Operating System Maniacs」

Far Northern Other World