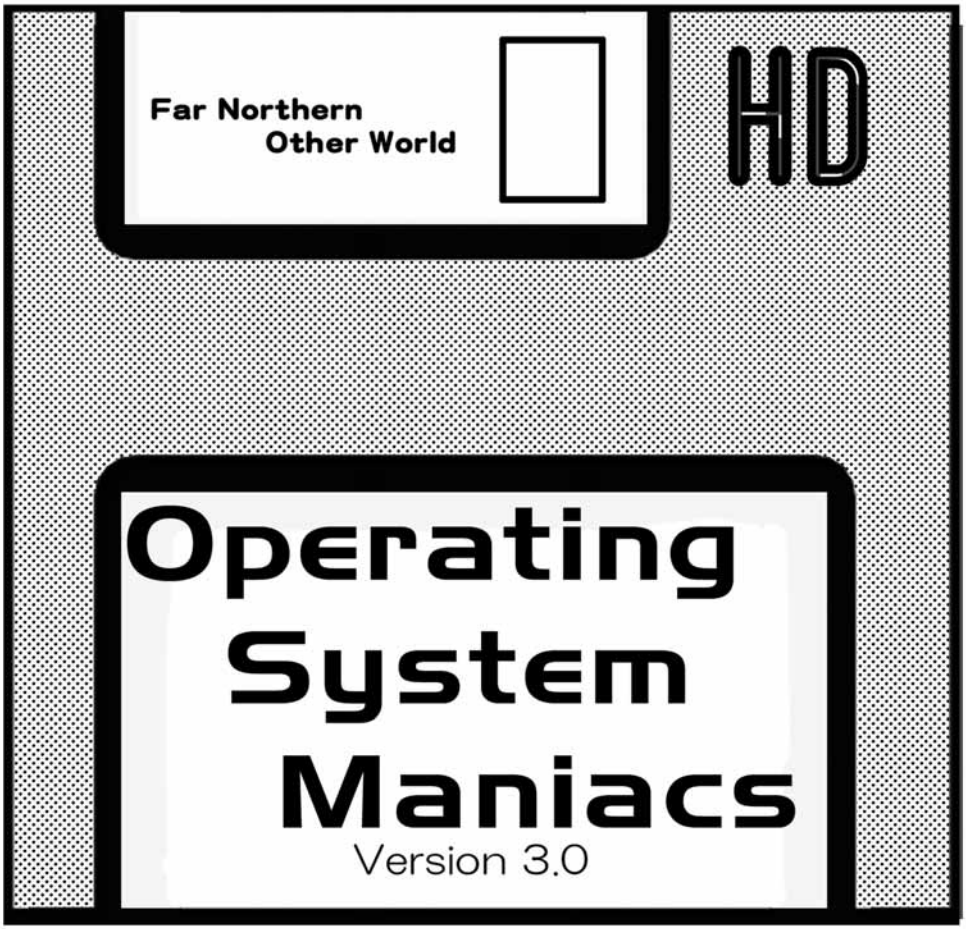


AELIX artasia asagao BARBUX blairOS BOS
BOZOS BRIX ContOS coron darkos DROPS FDOS
Freedos FRITZOS hanoi JANIX Idioma JxOS
knasos KOS Moubius QNX UUU PC-PTOS
MacOSX BOOTCAMP PS2Linux

Operating System Maniacs
Version 3.0
Now Booting.....

Far Northern Other World

Operating System Maniacs Version.3.0
BOOTLOADER

Licence Agreement？

Intel Macmini でマルチ OS/マルチブートの試み

りろ@涅槃

1 概説

Apple 社のマッキントッシュコンピュータは、本来 MacOS と一体として利用するものである。過去には Mac(PPC)用 Linux や BSD、そしてさらに昔は A/UX なる UNIX OS もあったが、あくまでも MacOS の動作マシンなのである。
Mac の最大の弱点は、高性能ゲームや安価なソフトウェアが少ないということであり、世は Windows ありきになっているから、熱心な Mac ユーザーであっても、ゲーム用の Windows マシンを別途購入しなければならない人も少なくなかった。

しかし、過去に何度も噂のあった Apple 社の Intel プロセサ採用の噂が、2005年に実現し、それならばデュアルブートで Windows も動くのではないかという期待が高まった。どこかのハッカーチームは、互換機(DOS/V)で MacOSX が動くようパッチを当てたメディアをヤミ配布していた。
そんなところへ、アップル純正のマルチブートツールである BOOT CAMP がリリースされ、Windows のライセンスさえあれば、もはや DOS/V 機を所有する必要はなくなったのである。
後で詳しく述べるが、BOOT CAMP が提供する機能は、次の2つである。
(1)起動ディスクのリパーティーショニング(空き領域のパーティション確保)
(2)Windows 用ドライバディスクの作成
つまり Windows とのデュアルブートに特化したツールなのである。
しかし、Mac が Intel プロセサを搭載した以上、様々な OS をマルチブートできるのではないかという期待が高まった。

かくして、BOOT CAMP やその他の手法を用いてマルチ OS を試みた実験を試みることとした次第である。

対象とする OS は、世間一般ではマイナーではあるが、マイナーOS 愛好家筋では十分メジャーな OS のみを対象とした。本当の超マイナーOS を対象にすると、1年以上の時間を費やすことになり、負担が大きすぎるからである。
動作しなかった OS の中には、インストールには成功したがカーネルレベルでエラーが出ているものも多いので、カーネル修正で対処できるものもあると思われるが、筆者にはそれだけの技術がないので断念した。
Mac でマルチブートを実現するには、次の3つの方法がある。

2 Mac でマルチブートを可能にする方法

(1)BOOT CAMP でマルチブート
MaxOSX10.5 Leopard で標準搭載されるツールであり、内蔵ドライブのリパーティショニングと Windows 用ドライバディスクを作成する。そのほか Windows インストールのガイダンスもしてくれる。このツールはあくまでも内蔵ドライブにパーティショニングをするためのものであり、外付けドライブには使えない。
(2)FireWire ブートによるマルチブート
Mac は古くから FireWire(IEEE1394)接続の外付けハードディスクからの起動をサポートしている。外付けディスク1基あれば、MacOSX の別環境を構築することも可能だし、ディスククローン機能によって内蔵起動ディスクの中身をクローニングし、サルベージ用環境も構築できる。もちろん Windows もインストール可能である(BOOT CAMP でドライバディスクを作っておく必要があるが
(3)USB ブートによるマルチブート
あくまでも MacOSX のみだが、ディスククローン機能で内蔵起動ディスクの中身をクローニングし、緊急時のサルベージ用環境を構築できる。
(その他の知識)
○BOOT CAMP が用意する Windows 用パーティションは、GUID パーティションである。よって GUID パーティションに対応していない OS はインストールできないか、できてもブートすることは難しい。
MacOSX のパーティショニングツール DISK UTILITY は従来型の MBR パーティションを作成することができるが、その場合 MacOSX と他 OS との共存は不可能である。(他 OS 同士の共存なら可能である。この場合は DOS/V 機でマルチブートを構成するのと基本的に同じことになる。
○上記の事情により、MacOSX と共存できる他の OS は極めて限られてくる。もしどうしても複雑なマルチブートを構成したいのであれば、外付け FireWire ハードディスクを用意し、ここに MBR パーティションを構築する必要がある。

3 事前準備

(1)実戦投入機を用いてテストするときには、必ず起動ドライブのバックアップをとること

ここでクローニングしたディスクは、外付け起動ドライブとして使えるので、万がーシステムを壊したときに復元が容易である。なお、ドキュメントフォルダの中身は日々、iDisk にバックアップを取るべきである。

(2)BOOT CAMP 動作時はすべての外付けハードディスク類の電源を切ること
BOOT CAMP は本来、内蔵ディスクに対してのみ有効であるが、各 OS のインストーラ(特に Windows と Linux)は、外付けハードディスクドライブを認識してしまうので、誤って関係ないドライブをフォーマットしてしまう恐れがあるからである。

(3)FireWire ブートを行う際には、そのディスクを MBR パーティションとして確保すること
Mac の基本は GUID パーティションテーブルであるが、現在これに対応している OS は少ない。MBR パーティションとして確保すれば互換機と同様に扱える

4 BOOT CAMP でマルチブート

(1)BOOT CAMP の基本操作
BOOT CAMP は、通常【アプリケーション】→【ユーティリティ】にインストールされている。

ここから、BOOT CAMP を起動すると、次のような画面になる

通常は、リパーティショニング画面になるのだが、筆者の Mac には既に Windows がインストールされているので、一気にこの画面になる。

2番目のタスクは、Mac Windows Drivers ディスクを作成することである。これは Windows インストール後に Mac の各種パーツのドライバを適用するためである。キーボードのドライバも適用されるが、仮名漢字変換まではサポートしてくれないので、注意が必要である。
ブランク CD-R を準備して、ディスクを作成する。(Leopard ではこの作業が不要になるという説がある。2007.10 現在)

この作業が終わったら、【Windows インストーラを起動】を選択し、Windows インストールメディアを挿入すると、リブートして CD から Windows インストーラが起動する。

Mac しか使ったことが無い人には、Windows のインストールになじみが無いと思うので、注意事項を書き留めておく。

Windows インストーラでは、ターゲットパーティションを指定する必要があるのだが、このとき「不明な領域」として表示されているのは、MacOSX のパーティションである。ここをターゲットにしてしまうと Mac 環境が破壊されるので細心の注意が必要である。必ず BOOT CAMP が確保した Windows 領域(C とか F とかのドライブレターが表示されているはずである)をターゲットに指定することである。

インストールが終わったら、再起動すれば Windows が起動する。Mac と Windows の使い分けは、option キーを押しながら電源投入ないしリブートすればディスク選択画面になるので、必要な方を選択する。
この後、Mac Windows ドライバディスクを適用する。無線 LAN やグラフィックアクセラレータが有効になり、コントロールパネルには Mac と同じ「起動ディスク選択メニュー」が表示される。

5 マルチ OS/マルチブートの結果(惨敗の記録)

本企画の中心、OS 別マルチブート対応記録である。Intelmac で MacOSX と Windows 以外にどの OS が動くのかテストした結果である。テスト用のマシンは Macmini Coreduo モデルである

OS 名	状況	敗因
FreeBSD	インストール成功　起動にも成功。ただしネットワークが使えず	ドライバがない
NetBSD	インストーラは正常起動するもキー入力を受け付けず	USB キーボード未対応?
OpenBSD	インストーラは正常起動するもキー入力を受け付けず	USB キーボード未対応?
PLAN9	CD-ROM 起動直後にインストーラ停止	原因不明
Fedora7	インストールは成功したが、ブート時にカーネルパニックで停止	カーネル固有の問題
SuSE Linux	Grub のインストールに失敗　起動せず	GUID パーティションであるにもかかわらず MBR に GRUB をインストールしようとした
Ubuntu	大成功(インストールも起動も全く問題なし)	
CentOS	インストールは成功。ただし起動せず	ブートローダーが行方不明?
KNOPPIX	LIVECD は正常動作	
Solaris10	STAGE2 でインストーラが停止	原因不明
UnixWare7	インストーラは正常起動するもキー入力を受け付けず	USB キーボード未対応?
NetWare7	ベースの DR-DOS のインストールに失敗	原因不明
QNX	インストーラ速攻で停止	とりつくシマもなし
超漢字	パーティションを認識せず	SATA ディスク未対応?

講評

(1)BSD 系
MacOSX のコアが BSD 系であるせいなのか、FreeBSD は数少ない成功例の一つである。ネットワークインタフェースのドライバが提供されれば、実用になりそうである。そして BSD チームの体質から考えて、Intelmac 初期モデル中古が出回る頃になったら「Macmini で作る FreeBSD サーバ」とかいうプロジェクトが立ち上がるのではないかと思われる。

(2)Linux 系
どのディストロもインストールは正常に終了するが、システムが起動しないケースが多く、成功したのは Ubuntu と KNOPPIX だけであった。GUID パーティションを正しく認識できるので、カーネルの修正再構築を行えば実用になると思われる。

(3)商用 UNIX 系
全滅である。なすすべもなし

(4)その他
超漢字は、SATA ディスクを認識できずに終了。
NetWare でベースとなる DR-DOS がインストールに失敗という結果であった。

6 エミュレータの活用

Mac によるマルチブートが惨憺たる結果に終わった以上、頼みの綱は PC エミュレータとなる。
かつて PowerPC の時代、PC エミュレータと言えば、Microsoft の VirtualPC しか選択肢が無く、しかもこの製品は到底実用に耐えるものではなかった。
しかし、IntelMac になってからというもの、Pararell Desktop と VMWare Fusion という2つの優れた製品が登場した。どっちを選ぶかは好みの別れるところだが、筆者としては VMWare Fusion を推薦しておきたい。
その理由
(1)対応 OS の種類が多い
(2)BOOT CAMP パーティションを起動できる
(3)64 ビット対応
本来、対応 OS のリサーチを行うべきだが、さしあたり簡単に述べると、OS/2 以外はほぼ問題なく運用できるので、省略させていただく。もしどうしても OS/2 またはその後継 eComStation を使用したい場合は、Pararell Desktop を選択すると良いであろう

7 そして私はマルチブートをやめました。

OS 別対応記録の章で見た通り、Mac でマルチ OS 環境を運用するのは惨敗という結果になった。様々な OS を試したいのであれば VMWare Fusiion の仮想マシンで運用が可能であり、その方が楽である。BOOT CAMP が必要となるのは、Mac で Windows 版高性能ゲームを楽しみたいという場合のみであり、その他の場合はおとなしく Mac を使っているのが正解といえる。

(以上)

PS2 Linuxをいじってみる ちょっとだけ後日談

立神楨一

1.イントロダクション

PS2 Linux導入後のあれこれ
前号で、PS2Linuxと、PlayStation BB Navigatorの共存インストールを試みました。
幸いにしてとりあえず実験は成功し、現在当方のPS2はデュアルブート？の形になっています。
さて、最近当方はいろいろとゲームを購入し、遊んでいますが、気になったことがありました。
「PS2はピックアップ部分が弱く、老朽化することがある」という情報がそれです。
そこで、PS2にはHDDがあるのですから、HDDにインストールして遊べるものは勿論、そうでないゲームもHDDから起動できないものなのかと考えたわけです。そうすればピックアップの老朽化も防げるでしょうし。
そもそも内部的にLinuxを使用しているわけですし、なんとかいじれないものかなぁと調べてみましたので、その辺を書いてみたいと思います。
ただし、最初に記載しておきますが、今回は実現に至っていません。理由については後ほど記載します。

2.ゲームをHDDにインストールするには

ちょっとGoogleで検索してみると、PS2へのゲームインストールツールがいくつか引っかかってくると思います。HDAdvance、HDLoader、などの名称がついているものがソレです。
また、今回試みようとしていることは、ヤフオクなり何なりで上記名称の製品を買えば、それだけで出来ることは出来るらしいです(当方は未検証)。無論本体のバージョンなどにもよりますが。
まずはこれらのソフトウェアは何をしているのか、という点を分析してみたいと思います。
また、すこし検索すると、ツールとしては大変グレーではあるものの、FreeHDLoaderといったような、フリーのHDDインストール用ソフトウェアもあるようです。ですが、これらのソフトウェアをインストールするには、結局メモリーカードを読み書きできるようなツール、ソフトウェア等が必要になるようです。
なお、ハード的に改造する方法もあるようですが、本稿の目的とは異なりますので、ここでは触れません。ご興味ある方は個別に検索等で調べてみてください。
さて、このソフトウェアを何とかPS2Linuxだけでメモリーカードにインストールできないものかというのが今回の原稿の趣旨です。ちょっと調べてみますと、どうも出来るのではないかと思われますので、手順部分だけではありますが、書いてみたいと思います。

3.実施手順の模索

今回の実施手法は、「PlayStation 2 Independence Day」と呼ばれる、起動BIOSの中のバッファオーバーフローのバグを利用して、ソニーからライセンスを受けていないコードをmodchip等のハードウェア改造なしでも動かす手法を実施してみるというものです。具体的には、PS2メモリカードのシステム設定ファイル内に仕掛けをします。
PS2は、PS1用のソフトを起動しようとする際に情報を mc0:/BIDATA-SYSTEM/TABLE.DB から読み取って起動するのですが、このファイルの読み込みにバッファオーバーフローのバグがあるために、TABLE.DBというDBファイルに
「mc0:/BIDATA-SYSTEM/BOOT.ELFを実行する」
というコードを混ぜておいて、起動を乗っ取るという仕組みです。
mc0はメモリカードのスロットで、指す位置により mc1 にもなりますが、これにも対応しています。
参考→http://hp.vector.co.jp/authors/VA008536/ps2linux/ps2independence.html
尚、本バグは、型番SCPH-50000の初期型あたりまで(基盤のリビジョンがVer9→Ver10になっているとのこと)残っており、以降の型番では解消されているそうです。この「PlayStation 2 Independence Day」を実施するためにメモリーカードに各種ソフトウェアをもぐりこませなければなりません。しかし、PS2Linuxからメモリーカードをマウントしてみても、上記のような
「BIDATA-SYSTEM」といったディレクトリは見当たりません。
じつはこれ、「ps2mcfs」というメモリーカード専用のファイルシステムの設定が「/BWLINUX/」をルートとして表示するようになっているからなのです。BBナビゲータからメモリーカード管理に入ってカードの中をみると、「あなたのシステム設定ファイル」というものがあります。これが、「BIDATA-SYSTEM」の正体です。
そこで、カーネルにパッチを適用した上で再構築し、メモリーカードの本来のルートディレクトリにアクセスできるようにします。

・・・が、ちょっと調べてみて今回、実験するにはどうも微妙な問題が内包されているらしいことに気がつきました。
BBN のカーネルが 2.4 系になっている所為なのか、2.2 系カーネルにパッチを当てたものではメモリーカードがマウントどころかフォーマット等が一切出来ないのです。
無論パッチの当て方が悪いとか、パッチ自体間違ってるとかもありえますが、実際に成功している方がいるということを考えると、PS2Linux(厳密には BB ユニット？)の Kernel が新しくなっているのでマウントできない可能性も出てきてしまいました。
いったん正規の 2.x カーネルで起動した上でメモリーカードをフォーマットし、その上でパッチを当てたカーネルで起動してみないとなんとも言えない状況になってしまいましたが、今回はそこまで時間が割けないので残念ながら、夏までは持ち越すことになってしまいました。。。値段も最近はこなれているようですしもう一台実験用に PS2 買うか。。。
一応、手順と必要なものだけ列挙します。今回はうまくいかなかったので、本手順で合っているかは検証できていないことにご注意ください。

1.必要なものの入手とカーネルの再構築
まず、以下のものを入手します。
kernel-source-2.2.21-pre1-xr7.tar.gz
http://playstation2-linux.com/からダウンロードできます。search から 2.2.21 などで検索すれば一発です。
bblinux2.diff.gz
http://hp.vector.co.jp/authors/VA008536/ps2linux/bblinux2.diff.gz からダウンロードできます。
PS2 形式の HDD パーティション認識パッチです。カーネルは BBN の(ReiserFS の)ファイルシステムしか認識しませんので、このパッチを適用します。
なお、これは前号でも書いた「怪しいパッチ」というものになります。違いは後ほど書きます。
no-bwlinux-check-2.2.21-pre1-xr7.diff
メモリーカード完全アクセスパッチです。
元々配布していたサイトはドメインが失効してしまったのか、現在このファイルは置いていません。
http://www.ocgnet.org/0xd6/
に移動しているようです。
カーネルの再構築を実施
cd /usr/src
mv linux linux.org
↑2.2.1_ps2 へのシンボリックリンクを待避。
tar zxf /var/tmp/kernel-source-2.2.21-pre1-xr7.tar.bz2　または　.gz
↑ダウンロード元に 2 種類置いてあります。どちらでも作業は可能です。圧縮形式の違いだけです。
ln -s kernel-source-2.2.21-pre1-xr7 linux
HDD フォーマット認識が BB Navi 仕様になっているので、PS2Linux 仕様に怪しいパッチを宛てる
cd /usr/src/linux
patch -p0 < bblinux2.diff
メモリーカード完全アクセスパッチを当てる
patch —p1 < no-bwlinux-check-2.2.21-pre1-xr7.diff
cd /usr/src/linux
cp -p config-xrhino .config　　→とりあえずデフォルトのカーネル設定にする
make menuconfig　　→オプションを好きに設定
make dep
make clean
make

make modules
make modules_install
カーネルは上記のとおり 2.2.21 を使用します。理由は「メモリーカード完全アクセスパッチが 2.2.21 までしかないから」です。また、もともとのカーネルに、メモリーカード完全アクセスパッチを適用するだけでもメモリーカードへのアクセスは出来るようですが、akload を使用しての DVD レス Boot と同時に実施する前提のため、上記の 2.2.21 を使用する手順にしています。
カーネルに精通している方がどなたか 2.4 用にパッチ書いてくれませんかねぇ。。自分でできれば言うことないのですが、どう

してもコード回りはなかなか・・・あるいは 2.4 だと出来ないとか？
あと注意ですが、今回の上記の手順で再構築するだけだと、起動時に BBN の起動途中で表示上は画面が止まった状態になるとか、そもそも前号の手順で行っている TELNET の仕込みとか akload の仕込とかいろいろ他にもしなきゃいけないことがありますが、今回はとりあえずの手順のみ記します。次号以後でまた検証できたらやってみたいと思います。

2.メモリーカードの BIDATA-SYSTEM を削除

メモリーカードの中の BIDATA-SYSTEM を削除します。
先ほども記載したとおり、BB ナビゲータからメモリーカード管理に入ってカードの中をみると、「あなたのシステム設定ファイル」というものがあります。これが、「BIDATA-SYSTEM」の正体です。
削除したら、起動時は別のメモリーカードを入れて起動してください。なぜかというと、起動時にささっているメモリーカードに、上記 BIDATA-SYSTEM は自動的に作成されてしまうからです。

3.akload から、先ほどのカーネルを使用して起動するように設定しなおす

現在は、最低限前号での共存インストールまでは終わっている環境として説明していますので、今回のカーネルを使用して起動すれば、BBN のパーティションをマウントできます。起動のさせ方やマウントの方法は前号のとおりですが、簡単に書きます。
起動については、現在メモリーカードから起動しているのであれば、メモリーカード内のカーネルを入れ替えてください。
http://hp.vector.co.jp/authors/VA008536/ps2linux/akmem_ps2.tar.gz をとってくる

```
# tar zxf akmem_ps2.tar.gz
# cd akmem_ps2
# make
# make mknod
# make install
# mount -t reiserfs /dev/hda11 /mnt/bbn
# cp -p /sbin/akload /mnt/bbn/sbin
```

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。

```
# cp -p /usr/src/linux/vmlinux /mnt/bbn/boot/ps2linux-2.2.21
```

「# 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-2.2.21


        /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2.2,21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。

4.起動しなおしてメモカをマウント
・・・とうまくいけばここでメモリーカードを確認する段階になるのですが、残念ながら今回は当方の環境ではそもそもメモリーカードがマウントできませんでした。
マウント失敗時のメッセージで調査をしてみたところ、どうも2.4系カーネルでないと動作しないものを2.2以前のカーネルで操作、アクセス等使用とすると出るメッセージであるという情報があったので(これが正しいかはわかりませんが)、もしかするとPS2Linuxのカーネル次第でメモカのマウントが出来たり出来なかったりする？可能性があるのかもしれません。
また、2.2系カーネルではメモリーカードへのアクセスやマウントを試みていないので、そもそも2.2系のデフォルト状態であればメモカがフォーマットできるかどうかも試していません。
そのため、今回の検証はココまでになってしまいました。
時間さえあればもうすこしつっこんだところまで検証できるかと思いますので、次回までの宿題としたいと思います。
一応、今回やろうとしていたことを箇条書き程度ですが記載しておきます。
BIDATA-SYSTEMフォルダを作成
TITLE.DBの中をのぞくとわかるのですが、起動の指定自体は実は「mc0:BRDATA-SYSTEM」となっていますが、これはそれぞれのregionのフォルダを勝手に見にいく仕様になっているようです。そのため、日本のPS2の場合は普通にBIDATA-SYSTEMフォルダを作成し、その中に必要なファイルを置けば良いようです。
そこに各種ハックされたファイルを置く。
さて、ここからは「PS2 HDD インストール ゲーム」でぐぐるといろいろ出てきます。そもそも、TITLE.DBというファイルを用意して別のものを読み込ませることは記載しましたが、そもそもどうやって別のファイルを読み込ませるのか、そして何を起動するのか、といったことはまったく書いてきませんでした。
実際のところ、使用するツールにはグレーなものもありますし、またホットな話題だったのはもう3年から4年ほど前のことです。
そのため、ここは若干簡素に書かせていただきます。ご興味ある方はご自分でググルなり、当方にご質問いただければ差しさわりの無い範囲でお教えいたします。
用意するファイル
titleman.zip
ps2mnu-k_v30.zip
HDLoder.elf
hdl_dumx-0.8.6.zip
簡単に書くと、titleman.zipでTITLE.DBを作成し、ps2mnu-k_v30.zipの中のBOOT.ELFを起動し、そのメニューからHDLoaderを起動し、HDDにインストールを行う、というものです。
今回はあまり実りの無い情報となってしまいましたが、次回はなんとかメモカの読み込み、書き込みについて試みてみたいと思います。
どうぞよろしくお願いします。
PS.
まだはっきりした情報ではない部分もありますが、上記のいくつかのHDDゲームインストールツールでは、単に購入した物をインストールしても外付けのHDDが認識できず、上手く行かないらしいです。そのためか上記の各ツールも、対応はSCPH-13000以降となっています。
ですが、一旦別のローダーなどで起動してから、HDDインストールツールを呼び出せば認識するとの情報も有りますので、その辺も含め継続調査していきたいと思っています。

〈了〉

PTOS と過ごした 80 年代

りろ@涅槃

(1)PTOS ヒストリー

私がまがりなりにも就職したのは、1985 年のことである。
私の入社した会社は、当時、ワープロ専用機として東芝 RUPO シリーズ、業務システムのホストとして、メインフレームの ACOS シリーズ、端末として NEC N5200 が使用されていた。取引先の中小企業では、オフコンサーバに NetWare、クライアントに NEC 9800(キュッパチ)シリーズを据えていたところも多かった。同じく取引先の印刷業者は Mac ありきでもあった。

1970 年代における IBM 対抗のための国産コンピュータ業界再編により、NEC と東芝が提携を組むこととなった次第は、日本コンピュータの歴史に関する参考書に譲ることとするが、営業的には、東芝 RUPO と NEC N5200 が霞ヶ関官公庁にシェアを広げたことにより、下部機関及び関連業者が一斉に右へならえをして、普及したようである。

搭載している OS は、PTOS(ピートス)であった。PTOS の正式名称は謎であるが、次のような特徴を持っていた。

〇完全な CUI OS であり、標準ではマウス等のポインティングデバイスを必要としない。
〇キーボードは基本的に JIS 仕様準拠であるが、PF キー(プログラマブルファンクションキー。JIS キーボードの F キーに相当)を21個備えて、各種機能を割り当てている独自仕様であった。
〇ウィンドウの概念がないので、一画面につき一つの作業しかできないが、その代わり画面切り替え機能を持ち、コマンドプロンプト画面1つとメニュー画面2つを切り替えられる。これによって理論的には同時に3つの作業が行えることになる。
〇ビジネス用デスクトップアプリケーションとして表計算ソフト LANPLAN データベースソフト LANFILE グラフ作成ソフト LANGLAPH(後に LANPLAN と統合され LANPLAN/G となった)ワープロソフト LANWORD を備えていた。
〇極めて使いやすいテキストエディタ(TEDIT)やマージ、ソートユーティリティを実装し、特に汎用機 ACOS との連携に優れていた。
N5200 は、当時「国民機」と呼ばれた NEC PC9800 シリーズと非常によく似たアーキテクチャを採用しており、N5200 が業務用パソコン、PC9800 が家庭用パソコンという区分けをされていた。しかし両者の価格差／性能差を見た場合、100 万円以上もする N5200 は、30万円程度の PC9800 に比べて分が悪く、90年代に入ると PC9800 シリーズで MS-DOS と PTOS のデュアルブートを可能にした PC-PTOS が発表された。
また、90年代中盤からのダウンサイジングブームに乗って、汎用機端末としての N5200 及び PC-PTOS マシンは次第に用済みになって行き、同時に Windows95 のリリースによって、時代はパソコン一色になっていった。あとに残ったのは膨大な LAN シリーズアプリケーション資産をどうするかであったが、これについては「自在眼」などのマイグレーションツールが有償で提供され、比較的スムースに Microsoft Office シリーズへの移行は完了した模様である。

(2)PTOS での業務処理

私が担当した業務システムは COBOL で記述され、汎用機 ACOS 上で動作する大規模統計処理システムであった。当時の大規模システムに特徴的なことであるが、汎用帳票を出力するシステムであり、別な視点での資料が必要になったら追加開発して新帳票を出力すると言うことの繰り返しである。いうまでもなくバックログは堆積する一方であり、気まぐれな上司からはすぐに欲しい資料が手に入らないことをあげつらって、「システム役立たず」と叱責される始末である。まだデータウェアハウスもない時代の話である。

しかし、システム運用人(私)まで無能呼ばわりされるのは不本意であったため、情報システム部門に依頼して、マスタの枢要データを切り出したワーキングデータをフロッピーディスク9枚に落とし、これを PTOS 上で動作する LANFILE を用いて集計するという手段を考案した。LANFILE はシーケンシャルデータを扱うデータベースソフトであり、当然検索もシーケンシャル総なめであるから、効率は悪かった。しかし、必要なキーでソートしておけば、原始的なクロス集計を可能にするという画期的な機能も実装していた。
LANFILE で集計した数値を、表計算ソフト LANPLAN に入力し資料の体裁を整えるという流れである。必要に応じてグラフ作成ソフト LANGRAPH も使用した。

この作業手順を考案したおかげで、誠に馬鹿馬鹿しいことであるが、私は社内では「コンピュータの神」と呼ばれるようになった。評価が上がるのは結構だが、単なる苦し紛れの知恵で考案した業務スタイルである。しかし本意に反して「コンピュータの神」という呼称は軛となり、ひたすらコンピュータのお勉強をさせられる羽目になった次第である。

(3)PTOS の限界と新システムの構築

しかし、便利というのはすぐに忘れられるもので、「これだけ便利ならもっと便利にしろ！」「俺が考えたことをボタン一つで資料出力するようにしろ」仕舞には「俺が考える前に何を考えるべきか計算するようにしろ」などと要望が相次ぎ、もう無茶苦茶である。HAL9000 ではないのだ。解決策は一つしか無い。幹部にデータベースを預ければよいのだ。かくして入力端末は PTOS 端末、中央処理系は ACOS6 資料作成は NEC EWS4800 で処理するシステム構築を担当することになった。その次第については、Operating System Maniacs Version2 に書いたので割愛するが、私のサラリーマン生活はまさに PTOS とともにあったと言える。

(4)PTOS の終わった日

NEC N5200 は、Windows 機が業務用として普及し始めた頃に対応して、徐々に役割を終えていき、1997 年頃には出荷停止となった模様である。PC9800 で動作する PC-PTOS については、N5200 資産の延命措置としてしばらく出荷が続いたが、業務システムの廃棄と再構築、LAN シリーズ資産のマイグレーションなどが進み、現在はほぼ絶滅状態である。

筆者は、2002 年頃に、何に使われているのか分からない N5200 マシンが事務室の真ん中に鎮座しているのを見たことがある。何に使うのか訪ねたら、給与計算に LANPLAN で作った表を使っているのだそうだ。速攻で Excel で再構築し、N5200 はすぐにお払い箱にした。無情なものである。

しかしながら、PTOS で仕事をしていたときには、Windows 機のように、しょっちゅう調子が悪くなることはなく、トラブルが起きた記憶もない。優れた国産 OS だったのだと思っている。

マイナーOS 駆け足レビュー

立神梢一

1.本稿の目的

マイナーOS は数多く有りますが、本誌では毎号、数種類を紹介するのが関の山です。
しかし、その沢山の OS の中には、いわゆる ToyOS と呼ばれるような物も数多く存在しています。
これら ToyOS は、OS としてユーザーが使用するというより、開発者が互いに参考にしたり、面白そうなプロジェクトに参加したり等の、どちらかというと開発者ベースで語られるべき物が多いと考えられます。
そう言った意味では、どちらかというとユーザーサイド、かつ開発言語系の知識がまるでない当方が扱うのは微妙なところもありますが、しかしながら、折角コレクションした OS ですし、起動や動作の実験が可能だった物について、スクリーンショットを交えて簡単にご紹介したいと思います。
尚、いくつか ToyOS というには発展している物、あるいは古い物とは言え、商用 OS や試用版も混じっています。(QNX のデモフロッピー等)
今回はそのうちの、主に1FD ベースで稼動するものについてレビューしてみたいと思います。次回があれば。。。いやあります。多分) HDD ベースのもの、LiveCD ベースのもの、などについても書きたいと思います。無論ここで紹介し切れなかったものについても。。。

2.本稿の環境

本稿は全て、以下の環境で行いました

2-1.物理的なマシンは用意しない。全て VMWare 上で行う

VMWare は Server 最新版をダウンロードして使用します。便利な世の中になったものですねぇ。
ちなみにカッコつけて Linux 版を使おうとしましたが、もろもろの理由で断念。ライブラリ周りが足りないのかうまくいきませんでした。

2-2.コンパイル等が必要な場合は、これも仮想環境上で行う

これは仮想環境上に Debian etch をインストールし、そこでコンパイルをするということです。
よくわからないライブラリの類を入れたりしなければならないケースもあるかなぁと思い、仮想環境内に用意してみました。

3.各項目の詳細について

基本的に、各 OS については以下のようにまとめています。

1. OS名
2. あればURL(オフィシャルサイトやsourcefargeなど)
3. スクリーンショット
4. 簡単な当方のコメントなど

という流れでご紹介していきたいと思います。
スクリーンショットは基本的にモノクロに変換しております。元がカラーのものなどは個別にコメントをつけたいと思います。
また、黒地に白文字で表示されるものが殆どですが、全て白黒を反転させています。黒ベタが多いと印刷にあまり綺麗に出ないと思われるためです。

各 OS のオフィシャルサイトはもちろんなのですが、現在のマイナーOS というか、ToyOS については、その殆どが
http://sourceforge.jp/ (日本)
http://sourceforge.net/ (米国)
を追っかければ大体のものは発見することが出来ると思います。
(いや、むしろ数が多すぎて完全に追いかけるのは不可能かもしれませんが、、、)
それでは、とりあえず当方が捕捉している OS のうち、とりあえずテストが出来た 1FDOS についてご紹介します。

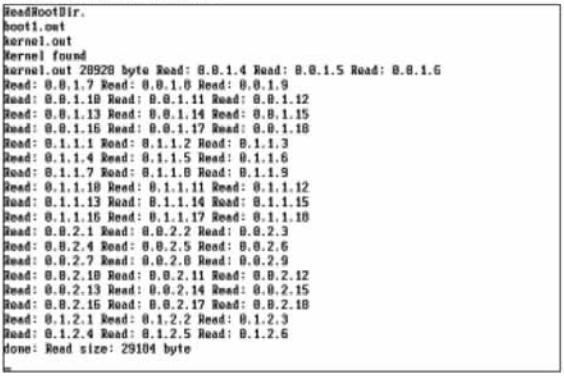
AelixOS

http://web.tiscali.it/aelix/

イタリアの方が作っていたらしく、イタリア語と英語のページがあります。
カーネルのみのようです。単にBootして画面に「AELIX」と文字を表示するしか機能がありません。
Alt+Ctrl+Delを受け付けて、再起動できます。(意外とキーボード入力を受け付けないOSもあるので、ちょっと以外でした)
2001年頃7月にリリースされて以後まったく動きがないようです。ただサイトは残っています

artesia

http://artesia.sourceforge.jp/



日本の方が作っているOSです。2004年頃で活動は止まっているように見えます。一人で開発していたらしく、あまり進行状況は芳しくなかったようです。起動して、BootDirの何かをカウントしています。(この辺はさっぱりなのでよくわかりません)。キーボード操作は受け付けず、操作等はできません。

あさがお OS

http://sourceforge.jp/projects/asagao-os/

これも日本人の方が作っていたOSです。起動すると、薄黄色の画面に青い文字で起動に成功したことを示す文字が出ます。2chの「OSをつくろう」スレへの謝辞も見えます。サイトURLは現在の一応の開発ページですが、リリース物件は登録されていません。一応、以下のURLからダウンロードは可能です。
http://wiki.osdev.info/?cmd=read&page=ScreenShot%2F%A4%A2%A4%B5%A4%AC%A4%AAOS

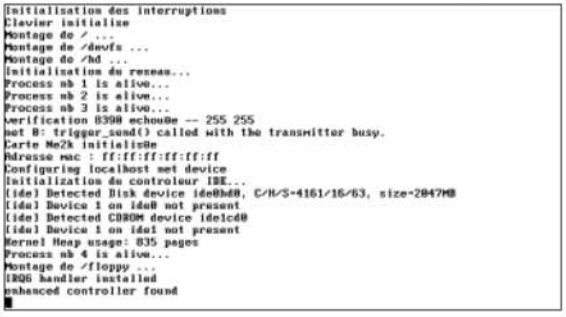
blairOS

http://sourceforge.net/projects/blairos/

2002 年にリリースがあって以降動きがありませんので、事実上ストップしているようです。起動するだけで、特に何が出来るわけではありません。。。。というかOSKitを利用して作成されているようですが、OSKitで作られたOSを起動すると一定の仕込まれた動作をした後、画像にもあるように_exit(0) called～と表示されて再起動を促されます。ってことは、多分これほとんど出来てないんじゃないかなぁ。。。
OSKitについては
http://www.media.osaka-cu.ac.jp/~k-abe/oskit/が詳しいです。

Barbux

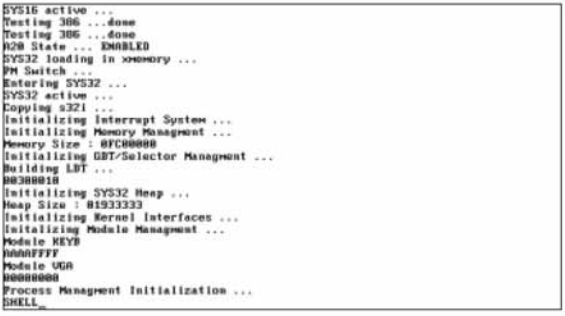
http://barbux.sourceforge.net/
http://sourceforge.net/projects/barbux/

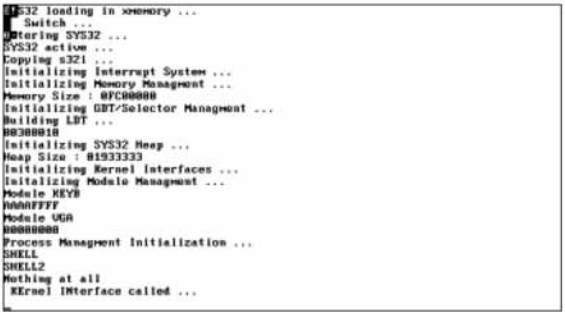


これも起動するのみのOSです。表示からすると、設定やコンパイルをちゃんとすれば、ネットワークも使えるのかも知れません。また、新しいものは一応IDEコントローラを認識しようとしているようです。これ以降のリリースが出ていないのでなんとも言えませんが。
あとオフシャルサイトが読めません。何語なんでしょうか。。。

BOS(Basic Operating System)

http://sourceforge.net/projects/bos/





Basic Operating System というだけあって、起動するだけです。
SHELL を呼びにいくのですが、作られていないために起動するだけの状態になっているようです。
尚、当たり前といえば当たり前ですが、SYSTEM/360 の顧客用に作られた太古の OS とは無関係です。
左上のところは赤や緑の色が付いていました。意味合いはよくわかりません。

BozOS

http://bozos.sourceforge.net/
http://sourceforge.net/projects/bozos/

これまた、SHELL を呼びにいくのですが、Shell が作られていないために結局起動するだけの状態になっているようです。
オフィシャルサイトを見ると、Kernel とファイルシステムとシェルを別個に作成しているようですが、シェルが無いようです。。

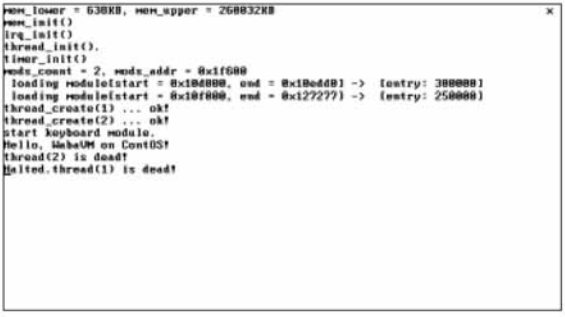
BRiX

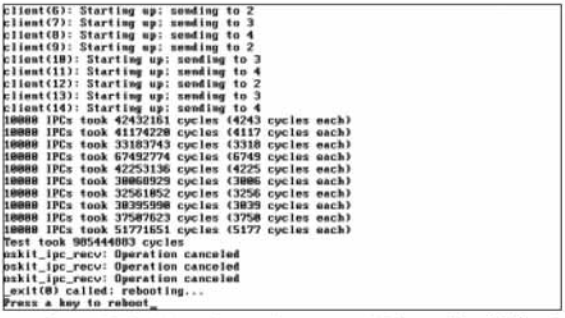
http://brix-os.sourceforge.net/
http://sourceforge.net/projects/brix-os/

これも起動して、なにやら計算をしているようです。
画面左上に延々チカチカと光っているのが何を意味しているのかよくわかっていません。。。。
ただ、これは今回のテストした OS の中では、ソースコードからコンパイルして動いた数少ない OS です。
他の動作確認をしたものは、殆どがソースと別に用意されていたバイナリ FD イメージを使用してテストをおこないましたが、その際にコンパイル自体は失敗するものも数多くありましたので、その意味ではちゃんとソースから make して動いたということで、うれしい部分はあります。

ContOS

http://tiki.is.os-omicron.org/tiki.cgi?c=v&p=ContOS

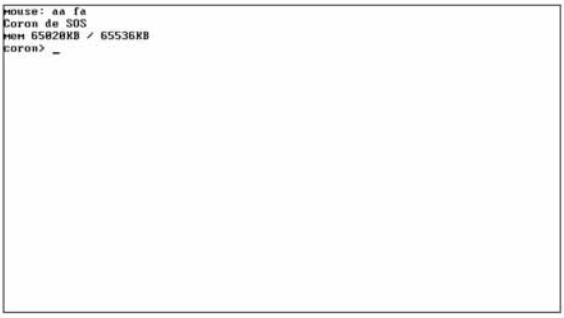


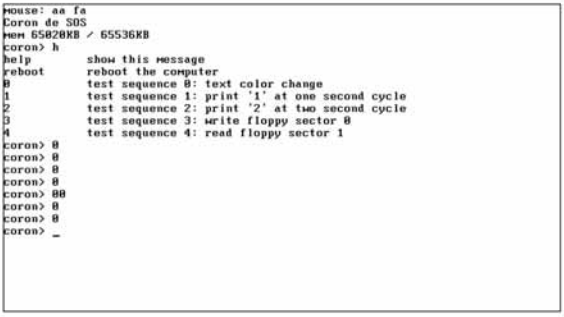


2002 年の 4 月頃にリリース(といっても ToyOS のようなので特に今後修正などはないのかな？)されています。これも当方の環境で make して一応動きました。
SoftwareDesign なんかでも記事を拝見する高野了成さんが開発しているらしいです。この方のページはいろいろ必見。開発畑で能力のある方ってのはすごいもんすね。。。

CORON

http://yike.mac.googlepages.com/coron





※2 枚目の画像は白黒にしてしまっているのでワケわかりませんが、フォント色の変更を試みています。

2002 年頃、LMKOS という冬休みの 2 週間で基本的な起動する OS を作ろうというプロジェクトが有り、それに成功したメンバが次に取り掛かったプロジェクトです。

2004 頃に FD ドライバを製作したところで開発メンバーの都合でプロジェクトはストップしてしまったようですが。。。。

```
coron> 0
coron> 0
coron> 4
wait: 0ms
intr_stat: c0h, 00h
--INIT:0
rec1
wait: 1ms
intr_stat: 20h, 00h
rec2
wait: 1ms
intr_stat: 20h, 00h
--OPEN:0
setup DMA channel 2
seek
wait: 1ms
intr_stat: 20h, 00h
read_id
wait: 0ms
intr_res: 04h, 00h, 00h, 00h, 01h, 06h, 02h
write d:c:h:s = 0:0:0:1
wait: 1ms
intr_res: 00h, 00h, 00h, 00h, 01h, 01h, 02h
--READ:0
coron>
```

```
intr_res: 00h, 00h, 00h, 00h, 01h, 01h, 02h
--READ:0
coron> 3
wait: 0ms
intr_stat: c0h, 00h
--INIT:0
rec1
wait: 2ms
intr_stat: 20h, 00h
rec2
wait: 1ms
intr_stat: 20h, 00h
--OPEN:0
setup DMA channel 2
seek
wait: 1ms
intr_stat: 20h, 00h
read_id
wait: 0ms
intr_res: 04h, 00h, 00h, 00h, 01h, 06h, 02h
write d:c:h:s = 0:0:0:1
wait: 0ms
intr_res: 00h, 00h, 00h, 00h, 01h, 01h, 02h
--WRITE:0
coron>
```

ただ、2007 年の 9 月に、これまでの成果物をオフィシャルにファイルにまとめてあります。立ち消えや開店休業になるプロジェクトが多い中、きちんと成果をあげた所までまとめてあるのはすばらしいですね。

DarkOS

http://members.fortunecity.com/darkqb/darkos.htm

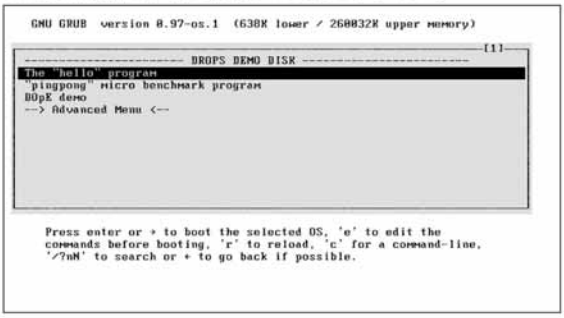
```
DarkBOOT
Loading.....1/0t_
```

ブートローダから起動するだけのようです。
動いたのか？と言われると微妙な所は有りますが、まあ起動まではしたのでよしとしますか。。。
ただ、サイトの記載によれば、機器によってはもうすこしちゃんと動くようです。時間が取れれば実機ででもテストしてみましょうか。

Drops

http://os.inf.tu-dresden.de/drops/
ドイツ・ドレスデン工科大学で開発されたマイクロカーネル「L4」を利用した、リアルタイム OS のプロジェクトです。
ちなみに DROPS は、"Dresden Real-time OPerating System"の略称とのことです。

DROPS 自体は複数のコンポーネントを合わせたものの名称で、現在のカーネル自体は上記 L4 とコンパチのカーネルである「fiasco」であるとのことです。
もともとは L4 カーネルで作成されていたものですが、L4 カーネル自体に何かしら問題があったためにコンパチである Fiasco カーネルを使っているとのことのようです。(98 年頃に Fiasco に移行したとのこと)
毎度のことで申し訳ないのですが Real-Time OS について詳しくないので、当該 OS で有効な実験が出来ていません。この OS 単体でも技術的なことにシフトすればもっと原稿も書けそうなものですが。。。
とりあえず、動作させてみた記録ということでご了承下さい。



Grub で Boot しています。

```
kernel (fd0)/l4/v2/bootstrap
   [Multiboot-elf, <0x100000:0xc000:0x218c>, shtab=0x10f118, entry=0x100000]
modaddr 0x02000000
Setting module load address to 0x2000000
module (fd0)/l4/v2/fiasco -noserial -nowait -nokdb
   [Multiboot-module @ 0x2000000, 0x521ac bytes]
module (fd0)/l4/v2/sigma0
   [Multiboot-module @ 0x2053000, 0x4e0c bytes]
module (fd0)/l4/v2/roottask
   [Multiboot-module @ 0x2058000, 0x2357c bytes]
module (fd0)/l4/v2/names
   [Multiboot-module @ 0x207c000, 0xa1a4 bytes]
module (fd0)/l4/v2/log --prio 0xA1 --buffer 8
   [Multiboot-module @ 0x2087000, 0x9304 bytes]
module (fd0)/l4/v2/dm_phys -v
   [Multiboot-module @ 0x2091000, 0x14544 bytes]
module (fd0)/l4/v2/l4io --noirq
   [Multiboot-module @ 0x20a6000, 0x4e1a4 bytes]
module (fd0)/l4/v2/l4dope --l4io -f
   [Multiboot-module @ 0x20f5000, 0x471a4 bytes]
module (fd0)/l4/v2/vscrtest
   [Multiboot-module @ 0x213d000, 0x67704 bytes]
vbeset 0x111
_
```

Kernel を起動しています。

```
  [00053000-00057130] (fd0)/l4/v2/fiasco -noserial -nowait -nokdb
  [00090000-00093658] (fd0)/l4/v2/sigma0
  [00094660-0009568c] (fd0)/l4/v2/sigma0
  [0009f000-00100000] BIOS area
  [00100000-0010e18c] Bootstrap
  [00124000-00141421] (fd0)/l4/v2/roottask
  [00142440-00201950] (fd0)/l4/v2/roottask
  [00300000-003986c8] (fd0)/l4/v2/fiasco -noserial -nowait -nokdb
  [01500000-015941a4] modules memory
  [01595000-015b057c] roottask syms/lines (copy)
  [0fef0000-ffffffff] deep space
  Starting kernel (fd0)/l4/v2/fiasco -noserial -nowait -nokdb proto=0x103000 at
0x0030093c


Welcome to Fiasco(ia32)!
DD-L4(v2)/x86 microkernel (C) 1998-2005 TU Dresden
Rev: Thu Aug 11 19:21:36 2005 compiled with gcc 4.0.2 for Intel Pentium
Performance-critical config option(s) detected:
  CONFIG_SCHED_RTC is on
  CONFIG_MDEBUG is off

Enabling special fully nested mode for PIC
Using the RTC on IRQ 8 (1kHz) for scheduling
```

Fiasco が L4 コンパチなカーネルであるためこのような表示になっていると思われます。

```
Roottask: Loading 1 module.
#05: loading "(fd0)/14/v2/pingpong"
     from [0157c000-015941a4] to [00000000-00015062][00016000-0096a000]
     entry at 000540ec via trampoline page code

Kernel version 01004444: Fiasco
Sysenter enabled
Rdtsc impact: 1-2: 112/454/941055 2-3: 214/321/20391, 1-3: 556/775/944041
CPU frequency is 1333Mhz, L4 Timer frequency is 1023Hz
32 MB scratch memory at 0c000000-0e000000 reserved

   0: short IPC intra address space      1: short IPC inter address space
   2: long  IPC inter address space      3: indrct IPC inter address space
   4: short fpage inter address space    5: long  fpage inter address space
   6: pagefault inter address space      7: exception intra address space
   8: short IPC inter c-inl-/asm-bind    9: compare fast sysenter IPC
   d: short IPC inter / don't switch     c: measure specific instructions
   m: memory bandwidth (memcpy)          i: short IPC inter AS, flooder
   s: change small spaces (now: 0-0)     a: all  x: boot  h: help  k: kdbg

>> 0: short intra IPC: 5.02 <==> 5.03
  === Shortcut (Client-Timeout(NEVER), Server-Timeout(0)) ===
     int30/warm: 1579601005 cycles / 100000 rounds >> 58745 <<
```

```
   8: short IPC inter c-inl-/asm-bind    9: compare fast sysenter IPC
   d: short IPC inter / don't switch     c: measure specific instructions
   m: memory bandwidth (memcpy)          i: short IPC inter AS, flooder
   s: change small spaces (now: 0-0)     a: all  x: boot  h: help  k: kdbg

>> 0: short intra IPC: 5.02 <==> 5.03
  === Shortcut (Client-Timeout(NEVER), Server-Timeout(0)) ===
     int30/warm: 1579601005 cycles / 100000 rounds >> 58745 <<
  sysenter/warm: 1932700931 cycles / 100000 rounds >> 62276 <<
     int30/cold:    9711166 cycles /     10 rounds >> 971116 <<
  sysenter/cold:    8735694 cycles /     10 rounds >> 873569 <<
  === No Shortcut (Receive Timeout 1s) ===
     int30/warm: 2867370719 cycles / 100000 rounds >> 106573 <<
  sysenter/warm: 2538963557 cycles / 100000 rounds >> 68259 <<
     int30/cold:    5399440 cycles /     10 rounds >> 539944 <<
  sysenter/cold:    5733142 cycles /     10 rounds >> 573314 <<

>> 1: short inter IPC: 6.00 <==> 7.00
  === Shortcut (Client-Timeout(NEVER), Server-Timeout(0)) ===
     int30/warm:  865790415 cycles / 100000 rounds >> 94557 <<
  sysenter/warm: 4134900168 cycles / 100000 rounds >> 84299 <<
     int30/cold:    2560710 cycles /     10 rounds >> 256071 <<
  sysenter/cold:    4103295 cycles /     10 rounds >> 410329 <<
  === No Shortcut (Receive Timeout 1s) ===
```

正直、ここらあたりはあまりよくわかっていませんが、とりあえず手当たり次第にキーボードから入力したりしてみています。

```
  sysenter/warm: 1932700931 cycles / 100000 rounds >> 62276 <<
     int30/cold:    9711166 cycles /     10 rounds >> 971116 <<
  sysenter/cold:    8735694 cycles /     10 rounds >> 873569 <<
  === No Shortcut (Receive Timeout 1s) ===
     int30/warm: 2867370719 cycles / 100000 rounds >> 186573 <<
  sysenter/warm: 2538963557 cycles / 100000 rounds >> 68259 <<
     int30/cold:    5399440 cycles /     10 rounds >> 539944 <<
  sysenter/cold:    5733142 cycles /     10 rounds >> 573314 <<

>> 1: short inter IPC: 6.00 (==> 7.00
  === Shortcut (Client=Timeout(NEVER), Server=Timeout(0)) ===
     int30/warm:  865798415 cycles / 100000 rounds >> 94557 <<
  sysenter/warm: 4134980168 cycles / 100000 rounds >> 84299 <<
     int30/cold:    2560710 cycles /     10 rounds >> 256071 <<
  sysenter/cold:    4103295 cycles /     10 rounds >> 410329 <<
  === No Shortcut (Receive Timeout 1s) ===
     int30/warm: 1702330933 cycles / 100000 rounds >> 102922 <<
  sysenter/warm:  707130332 cycles / 100000 rounds >> 92970 <<
     int30/cold:    3489277 cycles /     10 rounds >> 348927 <<
  sysenter/cold:    4305686 cycles /     10 rounds >> 430568 <<

>> 2: long inter IPC: 6.00 (==> 7.00 (sysenter)
  warm    4 dwords (   16B): 1597340423 cycles / 100000 rounds >> 144822 <<
  warm    8 dwords (   32B): 1586559317 cycles / 100000 rounds >> 144714 <<
_
```

```
>> 6: pagefault inter address space: 6.00 (==> 7.00 (sysenter)
  4kB => 4kB: 1398773165 cycles /  8192 rounds >> 169772 <<
  4MB => 4kB: 987689004 cycles /  8192 rounds >> 120557 <<
  4MB => 4MB:   1277703 cycles /     8 rounds >> 159712 <<

>> 7: intra exceptions: 5.02 (==> 5.03
  3240310099 cycles / 100000 rounds >> 32403 <<

>> 8: short inter IPC (c-inline-bindings): 6.00 (==> 7.00
     int30/warm:  362006329 cycles / 100000 rounds >> 89519 <<
  sysenter/warm: 3669166553 cycles / 100000 rounds >> 79641 <<
     int30/cold:    5075750 cycles /     10 rounds >> 507575 <<
  sysenter/cold:    9872969 cycles /     10 rounds >> 987296 <<
>> 8: short inter IPC (external assembler): 6.00 (==> 7.00
     int30/warm:  793371496 cycles / 100000 rounds >> 93833 <<
  sysenter/warm:  233203862 cycles / 100000 rounds >> 88231 <<
     int30/cold:    8304439 cycles /     10 rounds >> 830443 <<
  sysenter/cold:    5662319 cycles /     10 rounds >> 566231 <<

>> 9: short IPC intra-AS shortcut/no-shortcut: 5.02 (==> 5.03 (sysenter)
  sysenter/warm: 1258715515 cycles / 100000 rounds >> 55456 <<
  sysenter/warm: 1702352416 cycles / 100000 rounds >> 68773 <<
>> 9: short IPC inter-AS shortcut/no-shortcut: 6.00 (==> 7.00 (sysenter)
_
```

```
>> 8: short inter IPC (c-inline-bindings): 6.00 (==> 7.00
     int30/warm:  362006329 cycles / 100000 rounds >> 89519 <<
  sysenter/warm: 3669166553 cycles / 100000 rounds >> 79641 <<
     int30/cold:    5075750 cycles /     10 rounds >> 507575 <<
  sysenter/cold:    9872969 cycles /     10 rounds >> 987296 <<
>> 8: short inter IPC (external assembler): 6.00 (==> 7.00
     int30/warm:  793371496 cycles / 100000 rounds >> 93833 <<
  sysenter/warm:  233203862 cycles / 100000 rounds >> 88231 <<
     int30/cold:    8304439 cycles /     10 rounds >> 830443 <<
  sysenter/cold:    5662319 cycles /     10 rounds >> 566231 <<

>> 9: short IPC intra-AS shortcut/no-shortcut: 5.02 (==> 5.03 (sysenter)
  sysenter/warm: 1258715515 cycles / 100000 rounds >> 55456 <<
  sysenter/warm: 1702352416 cycles / 100000 rounds >> 68773 <<
>> 9: short IPC inter-AS shortcut/no-shortcut: 6.00 (==> 7.00 (sysenter)
  sysenter/warm:  937615682 cycles / 100000 rounds >> 95275 <<
  sysenter/warm: 1269905107 cycles / 100000 rounds >> 141540 <<

>> d: short inter IPC / don't switch to receiver: 6.00 (==> 7.00
     int30/call: 20359591 cycles /    200 rounds >> 101797 <<
  sysenter/call: 17737949 cycles /    200 rounds >> 88689 <<
Panic: deceiving ipc
Return reboots, "k" enters L4 kernel debugger...
_
```

```
     int30/warm:  362006329 cycles / 100000 rounds >> 89519 <<
  sysenter/warm: 3669166553 cycles / 100000 rounds >> 79641 <<
     int30/cold:    5075750 cycles /     10 rounds >> 507575 <<
  sysenter/cold:    9872969 cycles /     10 rounds >> 987296 <<
>> 8: short inter IPC (external assembler): 6.00 (==> 7.00
     int30/warm:  793371496 cycles / 100000 rounds >> 93833 <<
  sysenter/warm:  233203862 cycles / 100000 rounds >> 88231 <<
     int30/cold:    8304439 cycles /     10 rounds >> 830443 <<
  sysenter/cold:    5662319 cycles /     10 rounds >> 566231 <<

>> 9: short IPC intra-AS shortcut/no-shortcut: 5.02 (==> 5.03 (sysenter)
  sysenter/warm: 1258715515 cycles / 100000 rounds >> 55456 <<
  sysenter/warm: 1702352416 cycles / 100000 rounds >> 68773 <<
>> 9: short IPC inter-AS shortcut/no-shortcut: 6.00 (==> 7.00 (sysenter)
  sysenter/warm:  937615682 cycles / 100000 rounds >> 95275 <<
  sysenter/warm: 1269905107 cycles / 100000 rounds >> 141540 <<

>> d: short inter IPC / don't switch to receiver: 6.00 (==> 7.00
     int30/call: 20359591 cycles /    200 rounds >> 101797 <<
  sysenter/call: 17737949 cycles /    200 rounds >> 88689 <<
Panic: deceiving ipc
Return reboots, "k" enters L4 kernel debugger...

    --_exit----------------------------------------------EIP: f0010761
(6.00) jdb: _
```

```
[GENERAL] general debugger commands
  h                         Show this help screen.
  g                         leave kernel debugger
  Return                    show debug message
  Jc<color>                 set the Jdb prompt color, <color> must be:
                            nN: noir(black), rR: red, gG: green, bB: blue,
                            yY: yellow, mM: magenta, cC: cyan, wW: white;
                            the capital letters are for bold text.
  Jd(+|-)                   on/off Jdb output to VGA/Hercules console
  Jh                        set Jdb screen height
  JH                        detect screen height using ESCape sequence ESC [ 6 n
  Jo                        list attached consoles
  E(+|-)                    on/off enter jdb by pressing <ESC>
  ^                         reboot the system
  B[<lines>]/<str>]         show (last n lines of) console buffer/search

[INFO] information about kernel state
  l(r|p)                    show ready/present list
  i(1|2|4|p)<num>           in port
  o(1|2|4|a|u|m)<num>       out port, ack/(un)mask/ack irq
  k                         show various kernel information (kh=help)
  d[t<taskno>]<addr>        dump memory of given/current task at <addr>
  p[<taskno>]               show pagetable of current/given task
--- CR: line, SPACE: page, ESC: abort ---_
```

カーネルデバッガーの画面のようです。

```
  A(r|w)<addr>              read/write any physical address
  M(r|w)<addr>              read/write machine status register
  m<frameno>                show mapping database starting at page
  R(l|a<num>)               list IRQ threads, attach Jdb to IRQ
  bt[t<threadid>][<addr>]   show backtrace of current/given thread/addr
  r[<taskno>]               display IO bitmap of current/given task
  lt                        show enqueued timeouts
  u[t<taskno>]<addr>        disassemble bytes of given/current task addr

[MONITORING] monitoring kernel events
  O<number>(+|-)            on/off special logging event
  I(+|-|=|R(+|-)|T(+|-))    on/off/buffer ipc logging, on/off result, tracing
  IS(+|-)                   ipc without shortcut on/off
  IC(+|-)                   ipc with C fast path / IPC with Assembler fast path
  Ir(t|T|a|A|s|-)           restrict ipc log to (!)thread/(!)task/snd-only/clr
  P(+|-|=|R(+|-))           on/off/buffer pagefault logging, on/off result
  Pr(t|T|x|-)               restrict pagefault log to (!)thread/!thread/addr/clr
  U(+|-|=)                  on/off/buffer unmap logging
  Ur(t|T|x|-)               restrict unmap log to (!)thread/addr/clr
  M(+|-|=)                  buffer/off/buffer next period IPC
  C(l|r)                    show/reset kernel event counters
  T(P(+|-|k|u|<event>))     enter tracebuffer, on/off/kernel/user perf

[DEBUGGING] real debugging stuff
--- CR: line, SPACE: page, ESC: abort ---_
```

```
  IC(+|-)                   ipc with C fast path / IPC with Assembler fast path
  Ir(t|T|a|A|s|-)           restrict ipc log to (!)thread/(!)task/snd-only/clr
  P(+|-|=|R(+|-))           on/off/buffer pagefault logging, on/off result
  Pr(t|T|x|-)               restrict pagefault log to (!)thread/!thread/addr/clr
  U(+|-|=)                  on/off/buffer unmap logging
  Ur(t|T|x|-)               restrict unmap log to (!)thread/addr/clr
  M(+|-|=)                  buffer/off/buffer next period IPC
  C(l|r)                    show/reset kernel event counters
  T(P(+|-|k|u|<event>))     enter tracebuffer, on/off/kernel/user perf

[DEBUGGING] real debugging stuff
  b(i|a|w|p)<addr>          set breakpoint on instruction/access/write/io access
  b(-|+|=)<num>             disable/enable/log breakpoint
  br<num>(t|T|a|A|e|1|2|4) restrict breakpoint to (!)thread/(!)task/reg/mem
  S(+|-)                    on/off permanent single step mode
  L                         show last branch recording information
  Ld<taskno>                show LBT of specific task

[MISC] misc debugger commands
  t[<threadid>]             show current/given thread control block (TCB)
  t(+|-)                    show current thread control block at Jdb every entry
  t?                        show the corresponding thread id to a TCB address
  H<threadid>               halt a specific thread

(6.00) jdb: _
```

ドレスデン大学の WebSite は、記事内にも記載した L4Kernel、Fiasco、それ以外にも盛りだくさんの OS 研究のページがあります。技術者寄りの方もそうでない方も、OS に興味があれば是非のぞいてみていただきたいですね。

FDOS

http://sourceforge.net/projects/fdosx86/

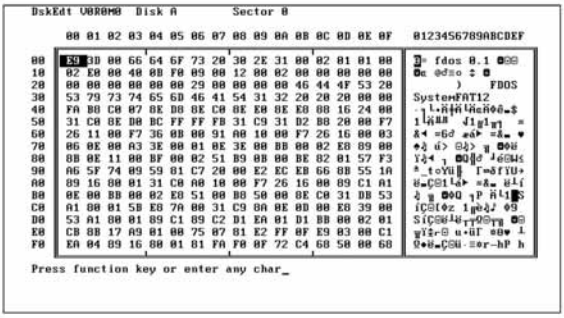
```
FDOS - Floppy Disk Operating System [Version: 0.0.9.4]
(C) Stefan Tappertzhofen 2002 - 2004



Deutsche Tastatur geladen.


A:\>_
```

```
Deutsche Tastatur geladen.


A:\>dir

  Datentraegerbezeichnung:
  A:

   FDOSX86                    0       Byte
   FDOS        SYS            11020   Byte
   DOS         SYS            304     Byte
   GAMES               [DIR]  0       Byte
   DOCS                [DIR]  0       Byte
   EXAMPLES            [DIR]  0       Byte
   STARTUP     BAT            0       Byte
   KEYB        BIN            1085    Byte
   DOSEML      BIN            399     Byte
   README      TXT            662     Byte
   RECYCLED            [DIR]  0       Byte

                              14998   Byte
                              1301504 Byte frei

A:\>echo readme.txt_
```

```
                              14998   Byte
                              1301504 Byte frei

A:\>cd examples

A:\EXAMPLES>dir

  Datentraegerbezeichnung:
  A:\EXAMPLES

   .                   [DIR]  0       Byte
   ..                  [DIR]  0       Byte
   EXAMPLE     BIN            255     Byte
   DIU0        BIN            167     Byte
   DSKEDT      BIN            1171    Byte

                              1593    Byte
                              1301504 Byte frei

A:\EXAMPLES>example

Hallo Welt - Hello World!
Random Number: 10310

A:\EXAMPLES>_
```

ドイツ製の DOS? FDOS です。

名称はまんま Floppy Disk Operating System だそうです。昔の DOS と混同しそうですね。。。



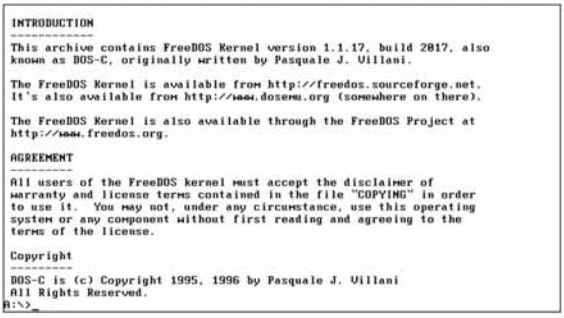
一応ピンポンゲームできたりします。左右のがラケットで、左のほうの点がボールです。

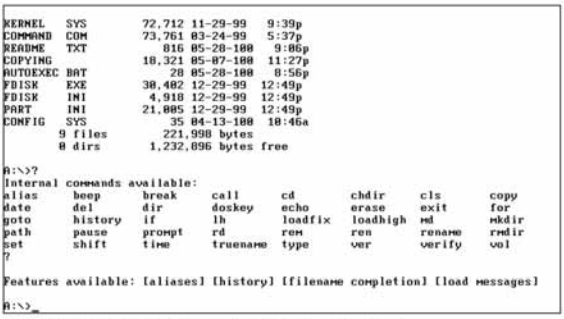
メッセージがドイツ語なんでよくわからない点もありますが。。。。
ゲームやプログラムから抜けられなくなったり、ベースがドイツ語キーボードなので英語に設定変更してやる必要があったりしますが、とりあえず OS としての体はなしている分そこそこの完成度と言えるのではないでしょうか。
終了時に、電源を切る状態にする、再起動する、などのオプションが選べます。
残念ながら現在は開発はストップしてしまっているようです。(最終リリースが 2004 年ごろ)

FreeDOS

http://www.freedos.org/

http://homepage1.nifty.com/bible/fdos/ （日本語サイト。FreeDOS/V）





知名度と完成度の割に今回扱いが小さくてすいません。

本 OS については可能であれば別途原稿を書いてみたいと思っています。QubeOS、SEAL といった上にのっかる GUI モノもあるので、ネタに困ることは無いでしょうし。

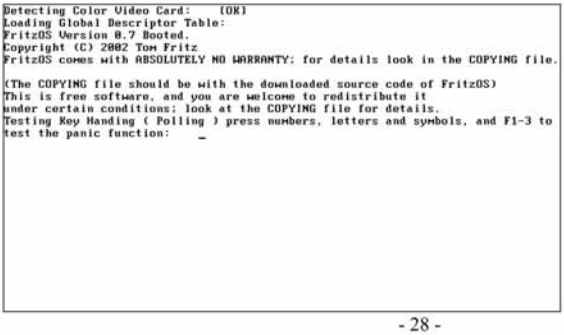
あまりきちんと調査したわけではありませんが、日本人の有志の方のサイトもありますし、DOS/V 化も不可能ではないようです。上記の日本語版公式？サイトから DOS/V 化されたものがダウンロードできるようです。ほぼ DOS コンパチで動くのではないでしょうか。

最近、本家がすこし迷走しているらしく(平たく言えば、いまさら DOS？ってことで迷走しているように見えます)、完成度の高いフリーの OS なのですから、何かしら方向性を見つけて頑張って欲しいですね。。。

上記にも記載した GUI ラッパーのレビューなどもしてみたいと思っていますので、そのときにまた。

FRITZOS

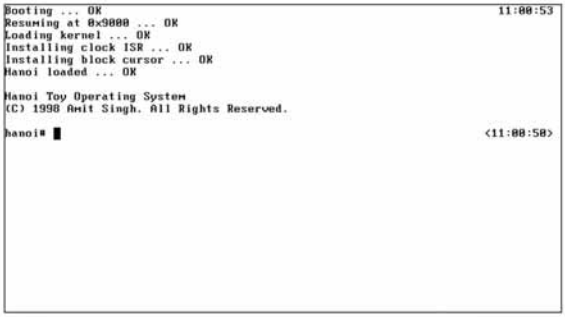
http://sourceforge.net/projects/fritzos/



とりあえず起動はしました。キーボードの認識は正常にしているようです。
F1 から F3 を押すことで、パニックを起こした後の状態で継続、リブート、Halt となるのですが、最初の2つはキーボードを押すと画面が変わってしまうのでキャプチャが取れませんでした orz

Hanoi

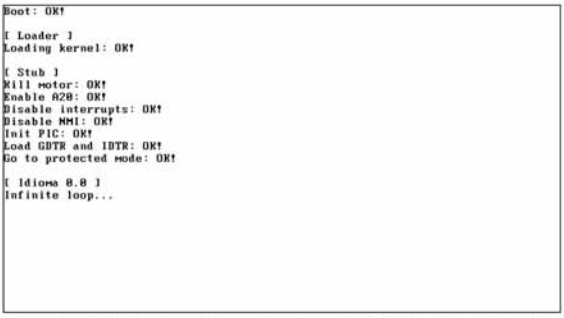
http://www.kernelthread.com/hanoi/html/tos-x86.html



キーボードを認識しますが、1 文字ごとに改行してしまいます。1 行目の右上のみ何故か水色でした。
直接 OS とは関係ないのですが、このサイト、http://www.kernelthread.com/mac/vpc/
で、大量の OS を Mac 上の仮想マシンで起動テストするというトンデモなことをしています。数がハンパじゃないです。正直この本の存在意義が危うくなるくらい(汗

Idioma

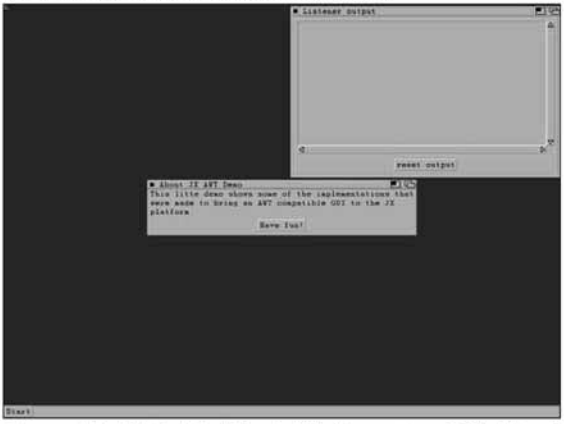
http://sourceforge.net/projects/idioma/



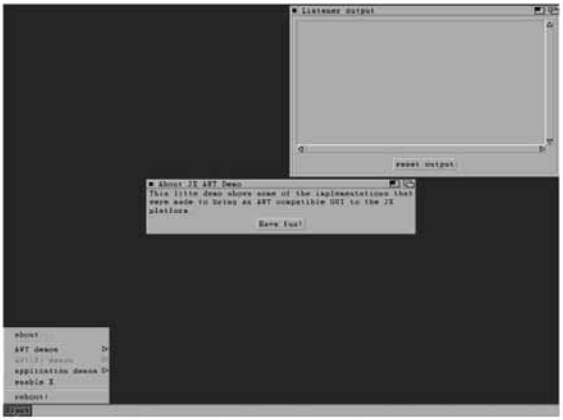
起動のみ確認しました。無限ループに入ってしまうようでそれ以上の動作はしませんでしたが。。。

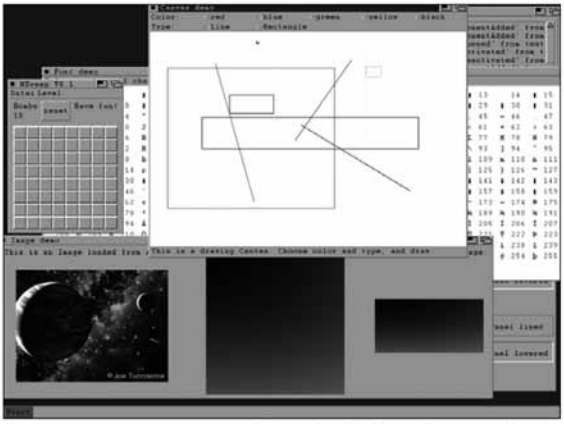
JxOS

http://www4.informatik.uni-erlangen.de/Projects/JX/



Erlangen 大学のドメイン内にあるので、同大学のコンピュータ科あたりの方が作成したものなのでしょうか。ちょっと同大学のサイトからのリンクはすぐには見つけられませんでしたが。
コピーライトの記載からすると、産学協同でなにか立ち上げてるのかも知れませんね。





Java ベースの OS です。かなり完成度は高く、簡単な直線と図形程度ですがペイントまがい のソフトが動い たり、マインスイーパ で遊んだり、その他もろもろのデモが動い たりします。無論マウス認識も問題なく行われました。
尚、今回は確認できませんでしたが、ネットワークを認識し、リモートデスクトップを使ったりも出来るようです。

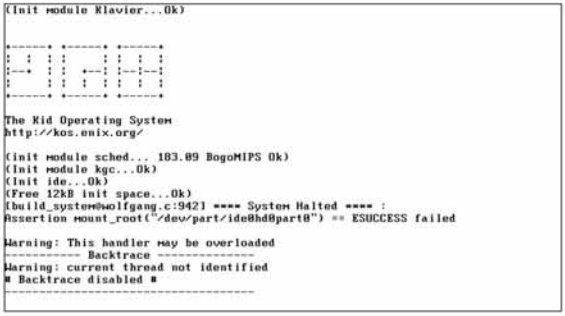
KnasOS

http://sourceforge.net/projects/knasos

一応当方で Build して、起動しました。また、いくつかコマンドも実装されているようで、した。といっても help でコマンド表示、あとは halt、beep、reboot、version といったコマンドのみですが。
Beep はもしかすると beep 音を鳴らすのかもしれませんが、ちょっと確認は出来ませんでした。
尚、全てアセンブラ言語で書かれているようです。あまり詳しくないのでアレですが、ソースコードの拡張子が全て asm でしたし、Makefile 見る限りは nasm しかコンパイルには使っていないみたいなので多分そうなのではないかと。

KOS

http://sourceforge.net/projects/kicer-kos/



Kid Operating System とのこと。
起動してすぐに Halt してしまいましたが、もしかすると実機を使ったりすればまた違うのかもしれません。
現在も開発しているようなのですが、よくわかりません。ちょっとサイト内をちゃんと追っかけられておらず、NEWS というところには 2005 年までしか記述がないのですが、一応ファイル等は 2007 年 9 月までは新規に UP されているのを見つけています。

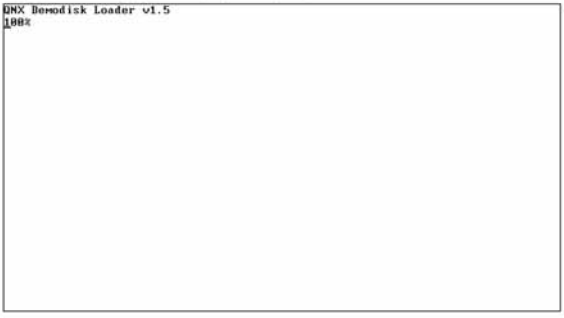
Moubius

http://mobius.sourceforge.net/
http://sourceforge.net/projects/mobius/

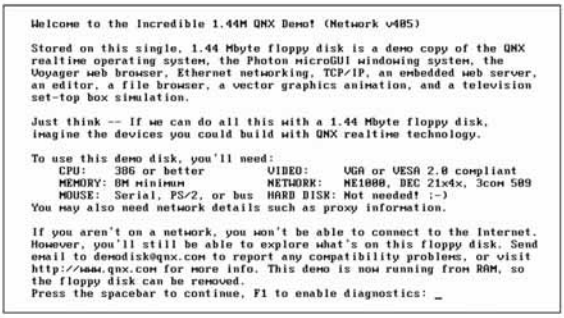
起動するだけです。その後フリーズしてしまっているようです。
サイトに掲載されているスクリーンショットによると、テトリスとかも出来るらしいのですが、うまくいきませんでした。可能性としては、やはりVMだからでしょうか。。。

QNX

http://www.qnx.com/
http://www.qnx.co.jp/
一号でも原稿を書いたQNXですが、これはその1FDD版です。Versionとしては QNX4 に当たるようです。発表された当時はそれなりに話題にもなったようです。1FDで、ブラウザまで動作する脅威のコンパクトさという論調だったようです。
http://journal.mycom.co.jp/special/2000/qnx/index.html
2000年の記事ですが、それなりにボリュームもあり、入門としては面白い記事です。
また、前号では6.3.xは無償利用できないようなことを書いたのですが、非商用、教育目的であれば無償利用できるようです。
上記のサイトから申し込み出来ます。
開発寄りの方にとってはさらに、今回QNXはソースコードへのアクセスについて無償で可能になったようです。ユーザー登録等が必要ではありますが、以下のサイトから色々と入手できるようです。
http://community.qnx.com/sf/sfmain/do/home

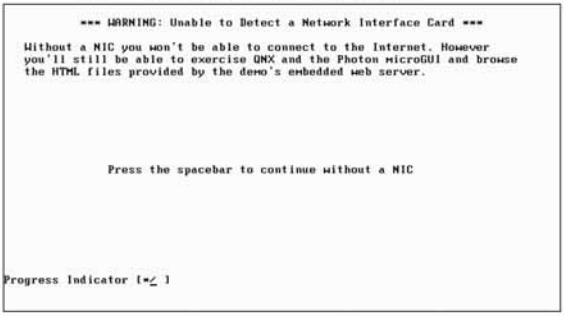


起動しています。以降3画面は青地に白文字で表示されています。

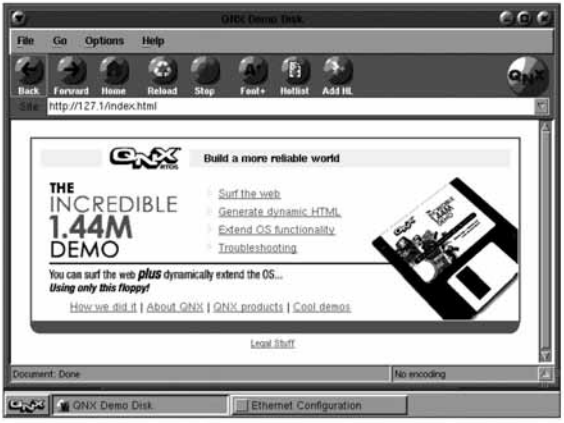


Welcome 画面とマシンの状態を示しています。ちなみに Network 用と Modem 用で別のディスクになっていました。一応、今でも探せばダウンロードは出来るようです。とりあえず解説付きでわかりやすい
http://qnx.projektas.lt/qnxdemo/qnx_demo_disk.htm
あたりが無難なところでしょうか。

ロケールの選択です。Asia がないのはまあ気にしない。

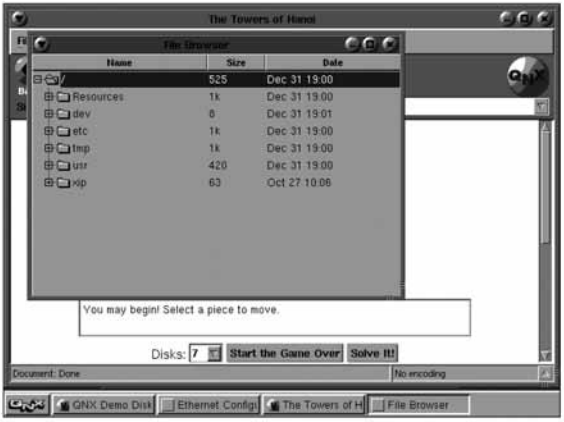


NIC が見つからないと言われています。VMWare だからだと思います。以前実機では正常に認識していました。今回はご紹介だけですので、ここは飛ばしてしまいます。



ちょっと見づらいですが、起動しました。ここからは GUI です。マウスも効きます。基本的に QNX はデスクトップにアイコンを表示するということはしないので殺風景です。

Javascript で書かれたゲームが出来るという DEMO です。

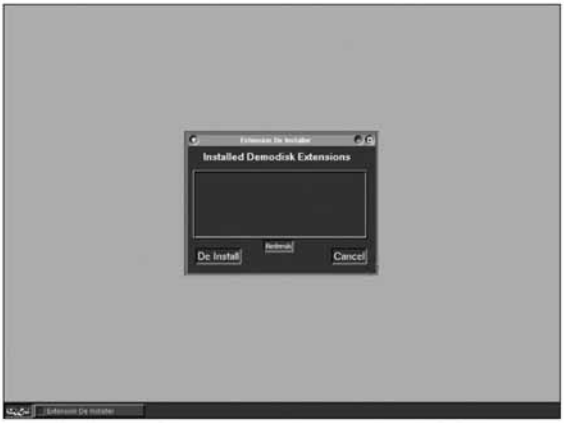


いわゆるファイルマネージャです。エクスプローラライクです。
かなりWindows的な操作を意識した作りになっています。
蛇足的ではありますが、前号で書いた6.2.xや6.3.xもそうなのですが、いわゆるWindowsのスタートメニューキーで、画面左下の「QNX」スタートメニュー(ランチャというべきか)が開きます。

テキストエディタのデモです。
文字が小さいですが、一応ちゃんと入力は出来ています。6.2.x以降はIMも(VJEですが)付いていますので、日本語入力もちゃんと出来ます。

カラーの文字がWindow内で動いているデモです。スクリーンセーバー的な動き方をしています。
色が濃いのでつぶれてしまっているかもしれませんが。。。

ちょっと見づらいですが、画像解像度の変更メニューです。
今回は変更できませんでしたが、こちらも以前実機でテストした際にはうまく動いていました。

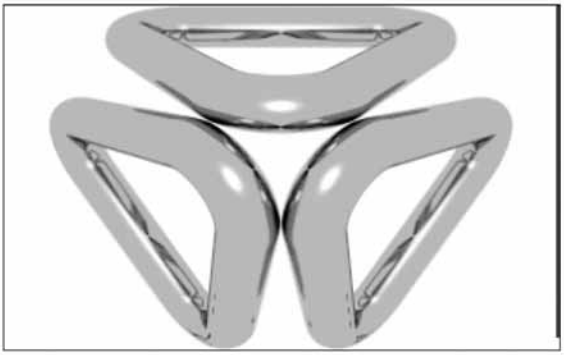


いわゆるアプリのインストーラです。
今回はちょっと試せませんでした。。。

かなり高機能かつ小容量ということで、いわゆるセットトップボックスなどへの利用が考えられていたようです。最近ではネットワーキング インフラストラクチャ、車載情報システム、医療機器、産業オートメーション、一般電気製品といったジャンルに使用されているとのこと。
HDD レコーダーなんかにも使われているようです。
BeOS の末裔が音響関連の OS としてひっそり使われていたり、こういった組み込み系の OS があちこちで使われていることを知ると、ますます OS について調べてみたくなったりします。

unununium

http://unununium.org/
http://sourceforge.net/projects/uuu/



```
Unununium stage 2 bootloader version $Revision$
run 'help' for a list of available commands.
stage2 > dir
command not found
stage2 > ls
command not found
stage2 > help
available commands:
echo - have the computer mock you
read-ata - read from the primary master ATA device
read-floppy - read sectors from the floppy
reboot - for compatibility with windows machines
thod - dump memory contents
stage2 > read-ata
USAGE: read-ata $ECTOR
stage2 > ecjp
command not found
stage2 > echo

stage2 > thod
USAGE: thod ADDRE$$ COUNT

Dumps COUNT bytes of memory at ADDRE$$. All numbers
are in hex with no leading 0x or trailing 'h'.
stage2 >
```

ウーンウーンウニュームとか読むらしいです。鉱石の名前？のようです。
カーネルをもたないとかいう開発方針だと言う話を見たことがあります。ヤドカリ動作前提ということなのでしょうか。
残念ながら開発者の方がモチベーションを保てなくなったようで、現状は開発は事実上ストップしています。一応リリースファイルは上記からダウンロードは出来ます。

4.雑記など

さて、超絶駆け足で 23 個の OS の簡易レビューをしてみましたが、いかがでしたでしょうか。
補足的に今回の実験中の出来事を書いておきたいと思います。

4-1.けっこう Build でできない OS が存在した

これはまあある程度は仕方の無いことかと割り切っています。無論、ソースコードに問題がある場合も合ったのでしょうけれど、物によってはリリース時期が古すぎて、今回当方の用意した環境には合わなかった可能性があります。

4-2.サイトなどでの説明以下の挙動しかしない OS(起動したらすぐ Panic とか)があった

これは、VMWare を使用して実験していることが大きいのかもしれません。理屈としては 4-1 と同様、やはりリリース時期が古いために、CPU 等の速度が速すぎる、IO 周りのドライバが合わない、等の理由があったものと考えています。

4-3.レビュー用のテストが OS の数に追いつかない

まあこれは仕方ない面もあるのですが、OS の収集ばかりしていた時期が相当あったため、テストがまるで追いついていません。もう数も数えていないくらいなので。。。
収集したころにはまだプロジェクトがあったのに、今回やっとテストしたらすでにプロジェクトはクローズ。。。なんてものもあまたあります。
まあ、活発なプロジェクトはそれなりに進んでいくでしょうし、当方は気の向くままにマイナーOS 道を突き進みたいと思っております。

いわゆるメジャー系およびそれに類する OS を抜かしても、まだまだたくさんの OS があります。事実、今回の原稿を進めていく中で、またさらに 20 個ほど見つけてしまったほどですから。(これでまた実験がおわらない。。。)
無論、そのほとんどは、冒頭でも述べた ToyOS です。機能がどうこう以前にまともに起動しないもの、そもそも Build 不可能なもの、あまりにもレガシー過ぎてテストできる環境がないもの。。。様々です。
しかし、それでもこれだけたくさんの方が各々のアプローチで OS を作ろうとしているというのは、やはり単純に機能だとかということとは別に、何かロマンのようなものがあるのかもしれませんね。

〈了〉

今回テストした OS の一覧

AELIX/artasia/asagao/BARBUX/blairOS/BOS/BOZOS/BRIX
ContOS/coron/Darkos/DROPS/FDOS/Freedos/FRITZOS
Hanoi/Idioma/jxOS/knasos/KOS/Moubius/QNX/UUU

今回 Build 不可だった OS の一覧(FD イメージが別途提供されていたため、実験は可能な OS も含む)

blinkOS/CEFARIX/CHARENGEOS/CONDUET/DELPHINEOS
FLICK/FUNIX/GAZOS/GEEKOS/guzOS/IANIXOS
Infinityos/Knossos/LMKOS

原稿募集要項

Operating System Maniacsでは、原稿を常時募集しています。

マイナーOSの導入/環境構築記事
異種アーキテクチャへのLinux/BSD系OSなどの導入/環境構築記事
その他業界動向、OSへの偏愛を吐露するコラム

我もと思う方は、是非ご参加ください。
詳細は、奥付連絡先までよろしくお願いします。

また、原稿ということでなくても、情報提供、レビューのリクエスト、素材の提供等も歓迎いたします。無論、記事内の間違い等についても、ご指摘いただけますと幸いです。

立神梢一

バックナンバー紹介

Version1.0(完売、在庫切れ中) VerionUP検討中
【悲運の正統UNIX】UNIXWAREを語る
SSS-PCを動かしてみる
【悲運の先端OS】OS/2～eComStation
QNXのインストールと環境設定
【マイナーOS】SkyOS Beの遺伝子を受け継ぐ「まぜこぜ」OS

Version2.0 在庫アリ
ChorusOSを動かしてみる
PS2LinuxをDVD-ROM無しで起動する
マイナーOS求む(研究したいマイナーOSの方向性について)
NEC国産OSへの鎮魂曲
OSとしてのVMWare ESX Server

全て頒布価格 300円 送料別

りろ@涅槃氏

Operating System Maniacs第3号に、またまた書かせていただき感謝です。
自分的には、OSという用語は使わず、「デスクトップシステム」と「サーバシステム」
という用語を使っていますが、それでも新しいOSが出ると試してみたくなるので、やっぱり自分もOSマニアなんだなぁと実感しています。
また次号も書かせていただければ幸いです。
次回は「超マイナーOSの実用化」という側面で書いてみたいと思います。

Writers Comment

立神梢一

.何とか3号も完成することが出来ました。
今回はPS2の追加テストが上手くいかなかったのは心残りですが、マイナーOSのそれなりの種類のレビューが出来たのは良かったです。
ページ数はともかく、ToyOS紹介はこれからも継続してやっていきたいと思っています。

編集後記と次号の構想

今回は実はページ数の都合ですこし当方の原稿を削りました。次号に掲載するヨテイです。(OS9000のテスト記事)
また、今回は夏→冬ということで時間があまり取れませんでしたが、その割にはまあなんとか原稿は揃えられたかなぁと思っています。
毎回原稿をお寄せいただいているりろ@涅槃氏には頭が上がりません。。。。

さて、マイナーOSネタはまだまだ沢山あります。
まだ確定事項ではありませんが、次号では以下のような原稿の構想を立てております。

1.ESを試す。
　任天堂が研究しており、今年の9月頃にOpenSourceとしてSourceforgeに登録されました。
　まだかなり敷居が高い段階のようですが、試してみたいと考えています。
2.PSLinuxネタの続き
　今回きちんとテストが出来ぬままでしたので、もう少し追跡調査が出来ればと思っています。
3.マイナーOSレビュー
　今回ほどの数、ページを割くかはわかりませんが、出来るだけ毎号、単独で記事を書くほどのキャパは無いけれど、紹介したいOSについて、ToyOSを中心に書いていければと思います。
それでは、またVersion4でお会いできればと思います。

お読みいただきありがとうございました。

立神梢一

Operating System Maniacs Version 3.0 奥付

発行日　2007年12月31日　コミックマーケット73　西ふ16a
発行　Far Northern Other World -極北別世界-
http://fnow.org
発行者　佐藤誠之(立神梢一)
印刷所　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
makoyuki@fnow.org

「Operating System Maniacs」

Far Northern Other World