注：旧 ConoHa で試したものです。現在の ConoHa ではまだ試せていません。

注:現在の ConoHa は、NetBSD をサポートしています (2015 年 11 月 18 日)。すばらしす。ConoHa で NetBSD のサーバーを建てようを見て下さい。

はじめに

ConoHa は格安で比較的使い勝手のいい VPS のひとつです。当然、NetBSD をインストールしたくなります。しかし、現状 (2015 年 3 月末現在) では、NetBSD 公式 (最新だと、NetBSD/amd64 6.1.5) の ISO イメージから boot できない状態です。

正確には boot はするのですが、即死します。

既知の回避策としては、VPS を 2 つつかい、PXE ブートさせて、インストールする、という方法があります。（2014 年の NetBSD BoF での Ryo ONODERA さんの発表が詳しいです。）しかし、インストールするためだけに、VPS を 2 つ借りてあれこれセットアップしないといけません。

もうひとつ回避策は知られていて、カーネルオプションとして、PCI_CONF_MODE=1、を加えたカーネルだと問題ありません。でも、そのようなカーネルで起動する、インストール用の ISO はありません。

この文書では、NetBSD のすぐれた環境構築のしくみをつかって、ConoHa のデフォルトのテンプレートイメージの CentOS 6.5 をつかって、ConoHa でつかえる、インストール用の ISO イメージをつくり、そのイメージで、NetBSD をインストールする方法を説明します。

残念なことに、最新のリリースである、NetBSD 6.1.5 では、PCI_CONF_MODE=1 のカーネルを作って試しても、そもそも boot すらしません。ここでは、NetBSD 7 Beta で行っています。

CentOS 側の作業

ConoHa の構成と CentOS 環境の準備

一番安い、VPS1GB プラン（メモリ 1G プラン）で試します。

最初にすることは、ディスク構成を変更することです。コントールパネルで、OS の再インストールを行います。そこで、構成を変えることができます。今回は、テンプレートイメージの CentOS 6.5 (64bit) を使い、ディスク構成はカスタム構成（100GB 全て利用する）こととします。

再構築にはしばらく時間がかかります。しばらく待ちます。

起動中になったら、ログインしましょう。ConoHa の Web コンソールは使い易い方ですが、別途 SSH で接続するコンソールや、直接サーバに SSH でリモートログインした方がよりいいかもしれません。やり方は、割愛します。

ConoHa のテンプレートイメージには、必要な開発環境がだいたい全て入っています。2015 年 3 月

末現在では、ひとつだけ足らないので、それだけ、yum install して下さい。

```
# yum install zlib-devel
```

NetBSD インストール ISO イメージの作成

■ git clone

root でログインしたら、NetBSD のソースコードを取ってきます。NetBSD の現在の安定版は 6.1.5 で、6 系が最新です。7 は、Beta の段階です。

NetBSD の公式リポジトリは CVS です。しかし、半公式なミラーとして、Github にリポジトリがあるので、そこから clone してきましょう。netbsd_7 という名前の branch です。

```
# cd /var/tmp
# git clone -b netbsd_7 --depth 1 https://github.com/jsonn/src.git
```

■カーネルオプションの追加

カーネルオプションを追加します。

NetBSD のソースコードはわかりやすい、と評されますが、機種に依存するカーネル関係のコードは ./src/sys/arch/ 以下にあるようです。x86_64 のことを NetBSD では、「amd64」と呼んでいますので、sys/arch/amd64/conf/ の下が、カーネルのコンフィグファイルの置き場所です。

ここに、GENERIC.local というファイル名で追加する値を書いたテキストファイルを置けば、GENERIC ファイルをいじらずとも、include されるようになっていますので、そのようにします。

```
# cd ./src/sys/arch/amd64/conf
# echo "options PCI_CONF_MODE=1" > GENERIC.local
```

■ build.sh 実行

あとは、NetBSD をコンパイルするだけです。

Linux 上で、NetBSD をコンパイルするのは、準備など大変な気がしますが、実は、src の直下に、build.sh、というヘルパースクリプトがあり、必要なクロスコンパイラなどの作成から、ビルド、ISO イメージの作成まで、コマンド一発で作成することができます。（NetBSD の大きな特徴のひとつです。）具体的には、build.sh -h でヘルプがでてくるので、見ていただければわかると思いますが、たとえば、

```
./build.sh -m amd64 release
```

で、amd64 用のリリースバイナリが作成できます。このあと、

```
./build.sh -m amd64 iso-image
```

で、ISO イメージが作成できます。

カンのいい方はおさっしのとおり、他のアーキテクチャでも同じです。たとえば、

```
./build.sh -m landisk release
./build.sh -m landisk iso-image
```

とすれば、NetBSD/landisk（IO-DATA 製の初期の NAS 等）用のインストール用 ISO イメージを作ることができます。

そもそも、NetBSD は、同じアーキテクチャ上でビルドする場合（つまり、amd64 上で amd64 をつくる場合）でも、コンパイラ等、NetBSD に必要なツールは全てクロス環境で用意され、その後、それらツールをつかって、クロスビルドされます。つまり、必要なツール群（toolchain といわれます）さえコンパイルできれば、NetBSD に限らず、かつ、同じアーキテクチャである必要もなく、より速いマシン上でビルドすればよい、ということが実現可能です。

ゆえに、CentOS 上で、NetBSD を簡単にコンパイルできるのです。

では、試してみることにしましょう。

```
# cd /var/tmp/src
# ./build.sh -m amd64 release
```

して、いきなりビルドしてもいいのですが、クロス環境がちゃんとつくれるかどうか、先に確認することにします。

```
# cd /var/tmp/src
# ./build.sh -N 0 -j 2 -m amd64 tools
```

ConoHa は、2 コアみえるので、-j 2 して、フルに使うことにします。（ConoHa の実力、見せてもらおうか ...。）

なお、-N 0 は、出力の抑制です。（build.sh のオプションは、./build.sh -h で確認することができます。）

```
<snip>
===> Tools built to /var/tmp/src/obj/tooldir.Linux-2.6.32-431.17.1.el6.x86_64-x86_64
===> build.sh ended:      Sun Mar 29 02:45:14 JST 2015
===> Summary of results:
        build.sh command:    ./build.sh -N 0 -j 2 -m amd64 tools
        build.sh started:    Sun Mar 29 02:39:04 JST 2015
        NetBSD version:      7.0_BETA
        MACHINE:             amd64
        MACHINE_ARCH:        x86_64
        Build platform:      Linux 2.6.32-431.17.1.el6.x86_64 x86_64
        HOST_SH:             /bin/sh
        MAKECONF file:       /etc/mk.conf (File not found)
        TOOLDIR path:        /var/tmp/src/obj/tooldir.Linux-2.6.32-431.17.1.el6.x86_64-x86_64
        DESTDIR path:        /var/tmp/src/obj/destdir.amd64
        RELEASEDIR path:     /var/tmp/src/obj/releasedir
            Updated makewrapper:  /var/tmp/src/obj/tooldir.Linux-2.6.32-431.17.1.el6.x86_64-x86_64
/bin/nbmake-amd64
        Tools built to /var/tmp/src/obj/tooldir.Linux-2.6.32-431.17.1.el6.x86_64-x86_64
        build.sh ended:      Sun Mar 29 02:45:14 JST 2015
===> .
```

このような表示が出れば、成功です。6 分くらいかかっていることもわかりますね。

では、いよいよ、本番です。

```
# ./build.sh -N 0 -u -j 2 -m amd64 release
```

-u は、アップデートオプションです。すでにビルドしたクロス環境を再度作り直さないようにするために、つけます。

```
<snip>
make release started at:  Mon Mar 30 07:39:19 JST 2015
make release finished at: Mon Mar 30 08:45:59 JST 2015
===> Successful make release
===> build.sh ended:      Mon Mar 30 08:45:59 JST 2015
===> Summary of results:
        build.sh command:    ./build.sh -N 0 -u -j 2 -m amd64 release
        build.sh started:    Mon Mar 30 07:39:18 JST 2015
        NetBSD version:      7.0_BETA
        MACHINE:             amd64
        MACHINE_ARCH:        x86_64
        Build platform:      Linux 2.6.32-431.17.1.el6.x86_64 x86_64
        HOST_SH:             /bin/sh
        MAKECONF file:       /etc/mk.conf (File not found)
        TOOLDIR path:        /var/tmp/src/obj/tooldir.Linux-2.6.32-431.17.1.el6.x86_64-x86_64
        DESTDIR path:        /var/tmp/src/obj/destdir.amd64
        RELEASEDIR path:     /var/tmp/src/obj/releasedir
            Updated  makewrapper:  /var/tmp/src/obj/tooldir.Linux-2.6.32-431.17.1.el6.x86_64-x86_64
/bin/nbmake-amd64
        Successful make release
        build.sh ended:      Mon Mar 30 08:45:59 JST 2015
===> .
```

一時間ちょっとかかりました。ConoHa さん、思ったより速いですね。

ISO イメージをつくります。

```
# ./build.sh -N 0 -u -m amd64 iso-image
```

```
<snip>
make iso-image started at:  Fri Apr  3 00:42:24 JST 2015
make iso-image finished at: Fri Apr  3 00:42:43 JST 2015
===> Successful make iso-image
===> build.sh ended:      Fri Apr  3 00:42:43 JST 2015
===> Summary of results:
        build.sh command:    ./build.sh -N 0 -u -m amd64 iso-image
        build.sh started:    Fri Apr  3 00:42:23 JST 2015
        NetBSD version:      7.0_BETA
        MACHINE:             amd64
        MACHINE_ARCH:        x86_64
        Build platform:      Linux 2.6.32-431.17.1.el6.x86_64 x86_64
        HOST_SH:             /bin/sh
        MAKECONF file:       /etc/mk.conf (File not found)
        TOOLDIR path:        /var/tmp/src/obj/tooldir.Linux-2.6.32-431.17.1.el6.x86_64-x86_64
        DESTDIR path:        /var/tmp/src/obj/destdir.amd64
        RELEASEDIR path:     /var/tmp/src/obj/releasedir
            Updated  makewrapper:  /var/tmp/src/obj/tooldir.Linux-2.6.32-431.17.1.el6.x86_64-x86_64
/bin/nbmake-amd64
        Successful make iso-image
        build.sh ended:      Fri Apr  3 00:42:43 JST 2015
===> .
```

これで、ConoHa で起動できる、ISO イメージができました。

なお、X 環境も build.sh で一緒にクロスビルドできます。必要な場合は、

```
# cd /var/tmp
# git clone -b netbsd_7 --depth 1 https://github.com/jsonn/xsrc.git
```

しておいて、

```
# cd /var/tmp/src
# ./build.sh -X /var/tmp/xsrc -x -N 0 -u -j 2 -m amd64 release
# ./build.sh -X /var/tmp/xsrc -x -N 0 -u -m amd64 iso-image
```

して下さい。

■ ISO イメージのアップロード

できた ISO イメージをアップロードしましょう。サーバから直接、といきたいところですが、残念なことに、CentOS 6.5 に付属の（OpenSSH 5.3 の）sftp では、private Key を指定するオプションがなく、サーバから直接アップロードできません。なんてこったい。

自宅等の回線が太いひとは、作った ISO イメージをダウンロードして、アップロードする方が速いと思いますが、ここでは、あくまでもサーバからアップロードしてみましょう。

private key が指定できる、より新しい OpenSSH をコンパイル、インストールします。

```
# cd /var/tmp
# yum -y install openssl-devel
# wget http://ftp.jaist.ac.jp/pub/OpenBSD/OpenSSH/portable/openssh-6.8p1.tar.gz
# tar xzf openssh-6.8p1.tar.gz
# cd openssh-6.8p1
# ./configure --prefix=/usr/local --without-zlib-version-check
# make install
```

これで、/usr/local/bin/sftp に、目的のものが手に入りました。

ConoHa のコントロールパネルから、アップロードに必要な SSH の private Key をいったん手元の PC 等にダウンロードして、サーバにアップロードします。ここでは、/root/.ssh/xxxxx.key にアップロードしたとします。

```
# cd /var/tmp/src/obj/releasedir/images
# sftp -i /root/.ssh/xxxxx.key xxxxxxxxxxxx@xxxxxxxx.cnode.jp
<snip>
sftp> ls
images
sftp> cd images
sftp> put NetBSD-7_BETA-amd64.iso
Uploading NetBSD-7_BETA-amd64.iso to /images/NetBSD-7_BETA-amd64.iso
sftp> quit
```

アップロードにかかった時間は 30 秒くらいでした。

これで、CentOS に二度と用はないので、shutdown します。

```
# shutdown -h now
```

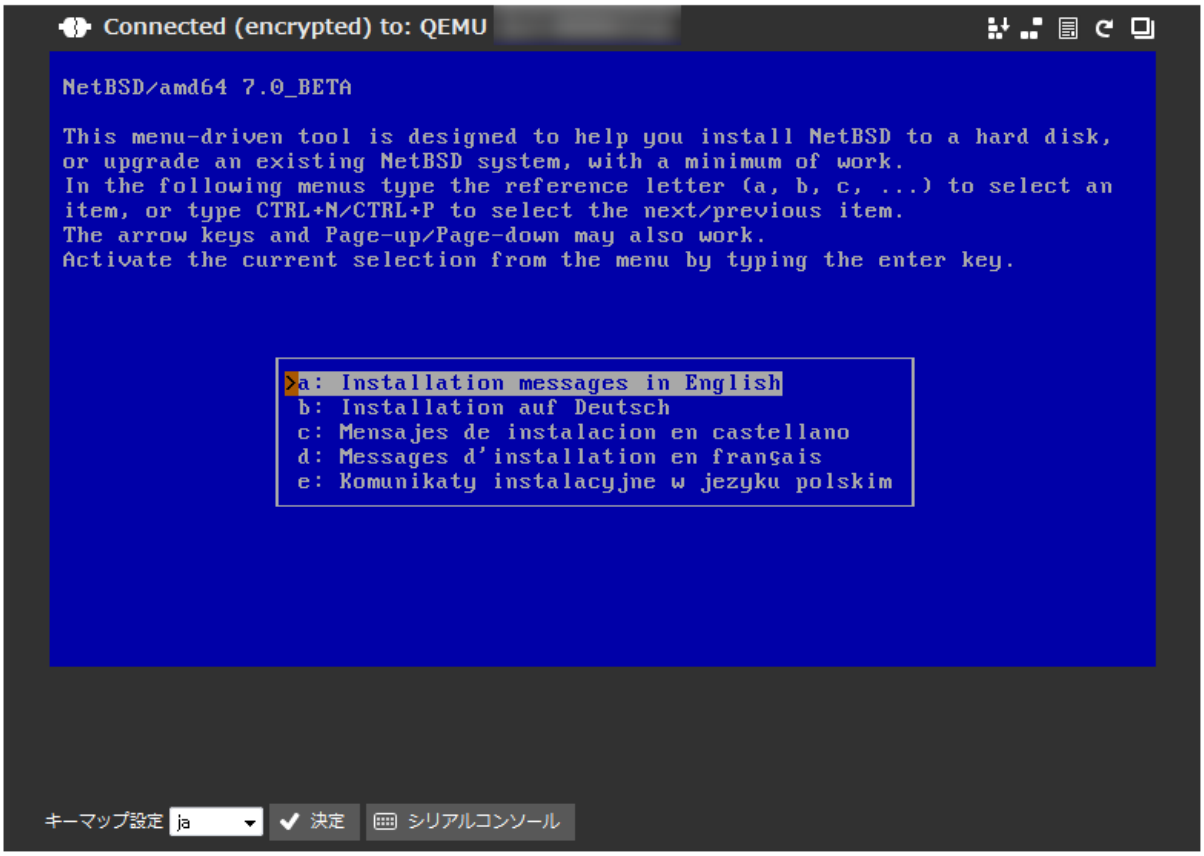
NetBSD 側の作業

あとは、任意の OS をインストール用 ISO イメージをつかって、ConoHa に入れるときと同じです。

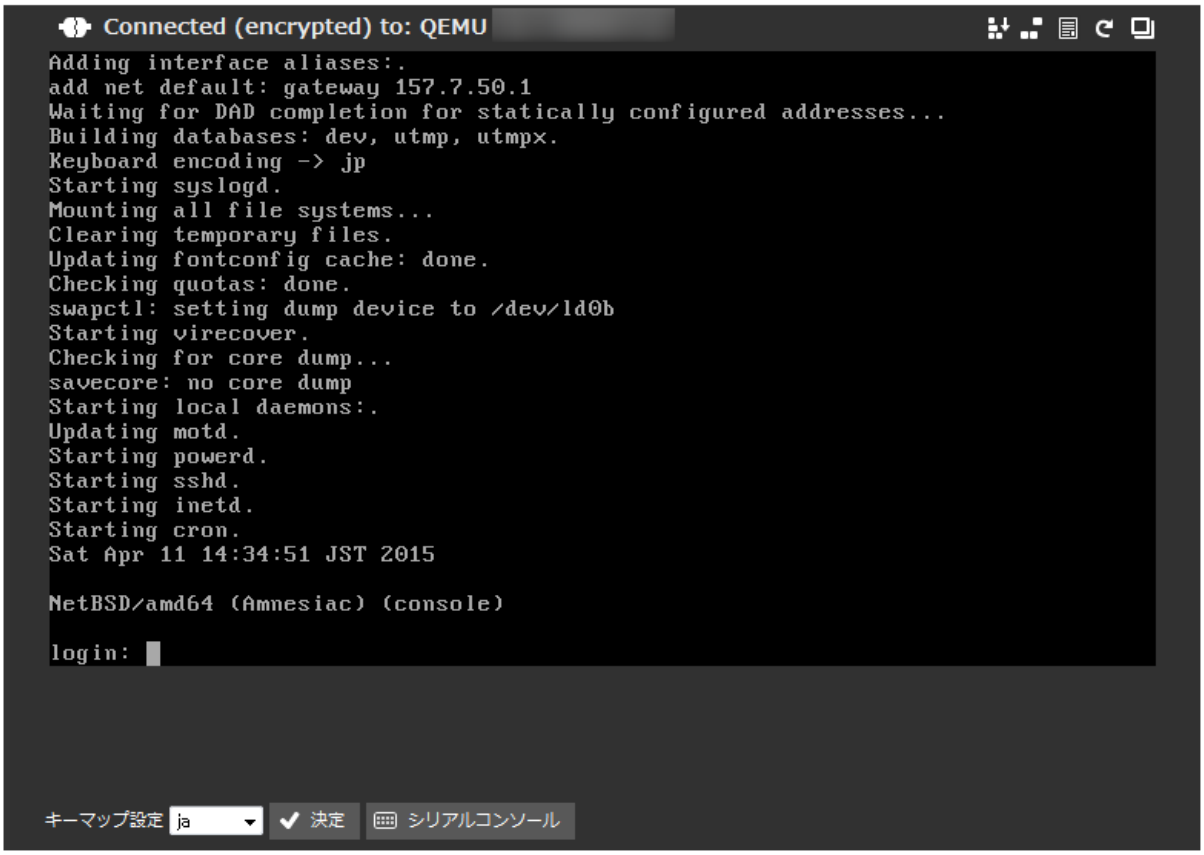
アップロードした ISO イメージが選択できるようになっているので、選択して「挿入」し、起動

して下さい。

無事起動できました。



あとは、インストールするだけです。インストールが終わって、再起動すると、



できました。

その他

dmesg

参考までに、dmesg です。

```
Copyright (c) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005,
    2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014
    The NetBSD Foundation, Inc.  All rights reserved.
Copyright (c) 1982, 1986, 1989, 1991, 1993
    The Regents of the University of California.  All rights reserved.

NetBSD 7.0_BETA (GENERIC) #1: Mon Mar  2 15:12:24 JST 2015
  sysbuild@ishitPc14.home:/opt/sysbuild/netbsd-7/obj/amd64/sys/arch/amd64/compile/GENERIC
total memory = 1023 MB
avail memory = 977 MB
kern.module.path=/stand/amd64/7.0/modules
timecounter: Timecounters tick every 10.000 msec
timecounter: Timecounter "i8254" frequency 1193182 Hz quality 100
OpenStack Foundation OpenStack Nova (2013.1)
mainbus0 (root)
ACPI: RSDP 0xfd900 000014 (v00 BOCHS )
ACPI: RSDT 0x3fffd740 000034 (v01 BOCHS  BXPCRSDT 00000001 BXPC 00000001)
ACPI: FACP 0x3fffff80 000074 (v01 BOCHS  BXPCFACP 00000001 BXPC 00000001)
ACPI: DSDT 0x3fffd9b0 002589 (v01   BXPC   BXDSDT 00000001 INTL 20100528)
ACPI: FACS 0x3fffff40 000040
ACPI: SSDT 0x3fffd8b0 0000FF (v01 BOCHS  BXPCSSDT 00000001 BXPC 00000001)
ACPI: APIC 0x3fffd7c0 00007A (v01 BOCHS  BXPCAPIC 00000001 BXPC 00000001)
ACPI: HPET 0x3fffd780 000038 (v01 BOCHS  BXPCHPET 00000001 BXPC 00000001)
ACPI: All ACPI Tables successfully acquired
cpu0 at mainbus0 apid 0: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz, id 0x306e4
cpu1 at mainbus0 apid 1: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz, id 0x306e4
ioapic0 at mainbus0 apid 2: pa 0xfec00000, version 0x11, 24 pins
acpi0 at mainbus0: Intel ACPICA 20131218
acpi0: X/RSDT: OemId <BOCHS ,BXPCRSDT,00000001>, AslId <BXPC,00000001>
LNKS: ACPI: Found matching pin for 0.1.INTA at func 3: 9
```

```
LNKD: ACPI: Found matching pin for 0.1.INTD at func 2: 11
LNKC: ACPI: Found matching pin for 0.3.INTA at func 0: 11
LNKD: ACPI: Found matching pin for 0.4.INTA at func 0: 11
LNKA: ACPI: Found matching pin for 0.5.INTA at func 0: 10
LNKB: ACPI: Found matching pin for 0.6.INTA at func 0: 10
acpi0: SCI interrupting at int 9
timecounter: Timecounter "ACPI-Safe" frequency 3579545 Hz quality 900
hpet0 at acpi0: high precision event timer (mem 0xfed00000-0xfed00400)
timecounter: Timecounter "hpet0" frequency 100000000 Hz quality 2000
pckbc1 at acpi0 (KBD, PNP0303) (kbd port): io 0x60,0x64 irq 1
pckbc2 at acpi0 (MOU, PNP0F13) (aux port): irq 12
FDC0 (PNP0700) at acpi0 not configured
COM1 (PNP0501) at acpi0 not configured
ACPI: Enabled 16 GPEs in block 00 to 0F
ACPI Exception: AE_NOT_FOUND, While evaluating Sleep State [\_S0_] (20131218/hwxface-646)
ACPI Exception: AE_NOT_FOUND, While evaluating Sleep State [\_S1_] (20131218/hwxface-646)
ACPI Exception: AE_NOT_FOUND, While evaluating Sleep State [\_S2_] (20131218/hwxface-646)
pckbd0 at pckbc1 (kbd slot)
pckbc1: using irq 1 for kbd slot
wskbd0 at pckbd0: console keyboard
pms0 at pckbc1 (aux slot)
pckbc1: using irq 12 for aux slot
wsmouse0 at pms0 mux 0
pci0 at mainbus0 bus 0: configuration mode 1
pci0: i/o space, memory space enabled, rd/line, rd/mult, wr/inv ok
pchb0 at pci0 dev 0 function 0: vendor 0x8086 product 0x1237 (rev. 0x02)
pcib0 at pci0 dev 1 function 0: vendor 0x8086 product 0x7000 (rev. 0x00)
piixide0 at pci0 dev 1 function 1: Intel 82371SB IDE Interface (PIIX3) (rev. 0x00)
piixide0: bus-master DMA support present
piixide0: primary channel wired to compatibility mode
piixide0: primary channel interrupting at ioapic0 pin 14
atabus0 at piixide0 channel 0
piixide0: secondary channel wired to compatibility mode
piixide0: secondary channel interrupting at ioapic0 pin 15
atabus1 at piixide0 channel 1
uhci0 at pci0 dev 1 function 2: vendor 0x8086 product 0x7020 (rev. 0x01)
uhci0: interrupting at ioapic0 pin 11
usb0 at uhci0: USB revision 1.0
piixpm0 at pci0 dev 1 function 3: vendor 0x8086 product 0x7113 (rev. 0x03)
timecounter: Timecounter "piixpm0" frequency 3579545 Hz quality 1000
piixpm0: 24-bit timer
piixpm0: LNKS: _SRS failed: AE_NOT_FOUND
interrupting at ioapic0 pin 9
iic0 at piixpm0: I2C bus
vga0 at pci0 dev 2 function 0: vendor 0x1013 product 0x00b8 (rev. 0x00)
wsdisplay0 at vga0 kbdmux 1: console (80x25, vt100 emulation), using wskbd0
wsmux1: connecting to wsdisplay0
drm at vga0 not configured
virtio0 at pci0 dev 3 function 0
virtio0: Virtio Network Device (rev. 0x00)
vioif0 at virtio0: Ethernet address fa:16:3e:3e:e2:20
virtio0: allocated 20480 byte for virtqueue 0 for rx, size 256
virtio0: using 8192 byte (512 entries) indirect descriptors
virtio0: allocated 81920 byte for virtqueue 1 for tx, size 256
virtio0: using 69632 byte (4352 entries) indirect descriptors
virtio0: allocated 8192 byte for virtqueue 2 for control, size 64
virtio0: interrupting at ioapic0 pin 11
virtio1 at pci0 dev 4 function 0
virtio1: Virtio Block Device (rev. 0x00)
ld0 at virtio1
virtio1: allocated 45056 byte for virtqueue 0 for I/O request, size 128
virtio1: using 36864 byte (2304 entries) indirect descriptors
ld0: 100 GB, 16383 cyl, 16 head, 63 sec, 512 bytes/sect x 209715200 sectors
virtio1: interrupting at ioapic0 pin 11
virtio2 at pci0 dev 5 function 0
virtio2: Virtio Block Device (rev. 0x00)
ld1 at virtio2
virtio2: allocated 45056 byte for virtqueue 0 for I/O request, size 128
virtio2: using 36864 byte (2304 entries) indirect descriptors
ld1: 0, 2 cyl, 16 head, 63 sec, 512 bytes/sect x 0 sectors
virtio2: interrupting at ioapic0 pin 10
virtio3 at pci0 dev 6 function 0
virtio3: Virtio Memory Balloon Device (rev. 0x00)
viomb0 at virtio3
virtio3: allocated 8192 byte for virtqueue 0 for inflate, size 128
virtio3: allocated 8192 byte for virtqueue 1 for deflate, size 128
virtio3: interrupting at ioapic0 pin 10
isa0 at pcib0
com0 at isa0 port 0x3f8-0x3ff irq 4: ns16550a, working fifo
attimer0 at isa0 port 0x40-0x43
pcppi0 at isa0 port 0x61
midi0 at pcppi0: PC speaker
```

```
sysbeep0 at pcppi0
fdc0 at isa0 port 0x3f0-0x3f7 irq 6 drq 2
attimer0: attached to pcppi0
acpicpu0 at cpu0: ACPI CPU
acpicpu0: C1: HLT, lat   0 us, pow     0 mW
vmt0 at cpu0: Unknown
vmware: open failed, eax=564d5868, ecx=0000001e, edx=00005658
vmt0: failed to open backdoor RPC channel (TCLO protocol)
acpicpu1 at cpu1: ACPI CPU
timecounter: Timecounter "clockinterrupt" frequency 100 Hz quality 0
IPsec: Initialized Security Association Processing.
uhub0 at usb0: vendor 0x8086 UHCI root hub, class 9/0, rev 1.00/1.00, addr 1
uhub0: 2 ports with 2 removable, self powered
uhidev0 at uhub0 port 1 configuration 1 interface 0
uhidev0: QEMU 1.0 QEMU USB Tablet, rev 1.00/0.00, addr 2, iclass 3/0
ums0 at uhidev0: 3 buttons and Z dir
wsmouse1 at ums0 mux 0
atapibus0 at atabus0: 2 targets
cd0 at atapibus0 drive 0: <QEMU DVD-ROM, QM00001, 1.0> cdrom removable
cd0: 32-bit data port
cd0: drive supports PIO mode 4, DMA mode 2, Ultra-DMA mode 5 (Ultra/100)
cd0(piixide0:0:0): using PIO mode 4, DMA mode 2 (using DMA)
Kernelized RAIDframe activated
pad0: outputs: 44100Hz, 16-bit, stereo
audio0 at pad0: half duplex, playback, capture
match_bootwedge: unable to read block 2049 of dev ld1 (22)
boot device: ld0
root on ld0a dumps on ld0b
root file system type: ffs
wsdisplay0: screen 1 added (80x25, vt100 emulation)
wsdisplay0: screen 2 added (80x25, vt100 emulation)
wsdisplay0: screen 3 added (80x25, vt100 emulation)
wsdisplay0: screen 4 added (80x25, vt100 emulation)
```

その他の *BSD は？

FreeBSD

公式対応しています。FreeBSD 10.1 が動きます。

OpenBSD

問題なく動きます。cf.https://twitter.com/tisihara/status/489780001726681089

DragonFly BSD

問題なく動きます。cf.https://twitter.com/tisihara/status/489419062599249920

MirOS BSD (MirBSD)

VirtIO を Off にすると、問題なく動きます。cf.https://twitter.com/tisihara/status/490186822299824128

Bitrig

インストール用の ISO イメージから起動できません ...。

上記以外は試していません。

注：現在の ConoHa は、OpenBSD をサポートしています (2015 年 12 月 2 日)。すばらしす。

謝辞

OSC の ConoHa ブースでもらった無償クーポンであれこれ試すことができました。ありがとうございました。> このはさん