Handhelds Mojo:
Building and running Ubuntu distributions on ARM

Andrew Christian Brian Avery George France

Handhelds Mojo:
Building and running Ubuntu distributions on ARM

Andrew Christian
Brian Avery
George France

The Historical Approach

Development computer

- Ubuntu/RedHat/Debian…
- >12,000 packages
- Native build environment

Mobile device

- Maemo/OpenEmbedded…
- ~700-2500 packages
- Cross-build environment

The mismatch between development and mobile device is a nuisance…

What we’d like…

Precompiled software for mobile & embedded devices with:

- Large numbers of up-to-date packages
- Well-defined releases with security and bug fixes
- Easy interoperability with the developer’s desktop
- Code compiled and optimized for our specific device

What’s the quickest way to get this?

- Compile the desktop distribution for the mobile devices!

Common concerns

Aren’t mobile devices too small to run desktop software?

- 256 MB of RAM and an SD flash card is larger than the laptops we used just 9 or 10 years ago

The graphical user interface for the desktop doesn’t make sense on a mobile device!

- GUI applications represent only a small number of the applications in a distribution.
- Many desktop distributions contain small-device applications (e.g., Mobile Ubuntu)

Quick Summary

- The Mojo project has been rebuilding Ubuntu distributions for different flavors of ARM processors

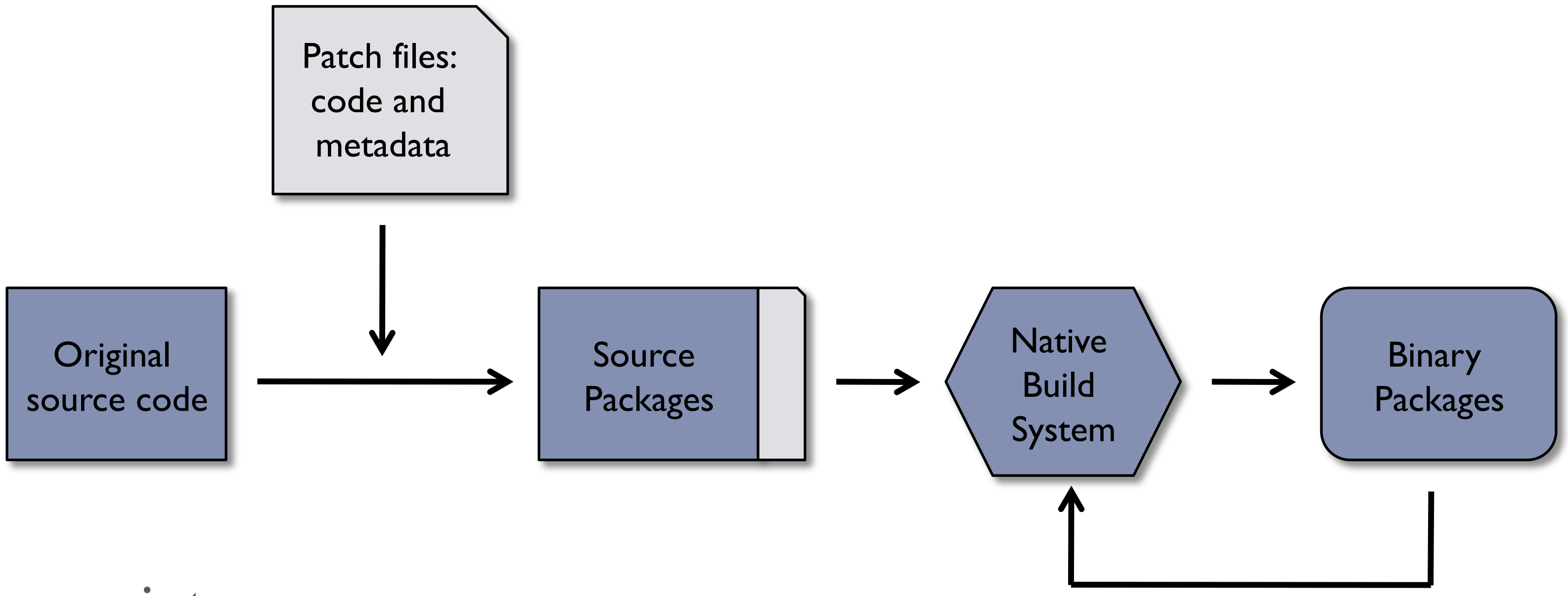
				ARM Targets	
Mojo	**Ubuntu**	**Released**	**v5**	**v5+VFP**	**v6+VFP**
Frisky	7.04	March 2008	✔		✔
Grumpy	7.10	June 2008	✔		✔
Hasty	8.04	July 2008	✔	✔	✔
Icy	8.10	Nov. 2008	✔	○	○

http://mojo.handhelds.org

Outline

- The challenges in building the distributions
  - Compilers, libraries, and toolchains
  - Native machine clusters
- The current state of the distributions
  - What works, what has been patched, what is missing
- How to use a Mojo distribution
  - Sample installation
  - Examples of systems that use the distribution
  - Performance
- Future work

Desktop distribution build process



Key points

- The build system is running its own packages. Iteration required!
- The build system runs on native hardware
- The toolchain is intrinsic to the distribution and gets compiled along with all of the other packages

Challenge: Toolchains

A toolchain is the combination of:

- C compiler (gcc)
- Linking and object tools (binutils)
- Standard C libraries (glibc)

You can’t build a distribution without a good, stable toolchain. But you can’t build the toolchain without a matching distribution….so you iterate.

Ubuntu’s toolchains

- The quality of ARM code produced and the number of architectures supported have generally improved over time.

	gcc	binutils	glibc
Dapper	4.0.3-1	2.16.1.cvs2006…	2.3.6-0ubuntu20
Edgy	4.1.1-6ubuntu3	2.17-1ubuntu1	2.4-1ubuntu12
Feisty	4.1.2-1ubuntu1	2.17.20070103…	2.5-0ubuntu4
Gutsy	4.1.2-9ubuntu2	2.18-0ubuntu3	2.6.1-1ubuntu9
Hardy	4.2.3-1ubuntu3	2.18.1~cvs2008…	2.7-10ubuntu3
Intrepid	4.3.1-1ubuntu2	2.18.93.2008…	2.8~20080505-…

Verifying you have a good toolchain

A "good" toolchain is one that passes most of its test suites.

- ARM is not the most popular architecture: building a "good" ARM toolchain requires a fair bit of testing and patching
- Toolchains depend in surprising ways on all sorts of other packages (e.g. Perl, bash, …)
- Number of errors from test suite decreases as you iterate; for example, for gcc 4.1.2, we went from 11 to 5 to 0 with each iteration.

To maximize distribution quality, we iteratively compile each distribution at least 3 times

What, exactly, do you compile for?

Compiler option	Choices	What it affects
Application Binary Interface	Old ABI, EABI (1-5)	Data structure alignment, how parameters are passed to/from functions, kernel interface
Floating point	Hardware, Software, Vector (VFP)	Format of floating point numbers and execution speed
Endian	Little/Big	How words are stored in memory
ARM Architecture	v3, v4, v5, v6, v7…	The instruction set
Thumb	Non-thumb/Thumb (& version)	Code size, execution speed, & interoperability
Target processor	E.g. Xscale	Optimization for a specific processor or family

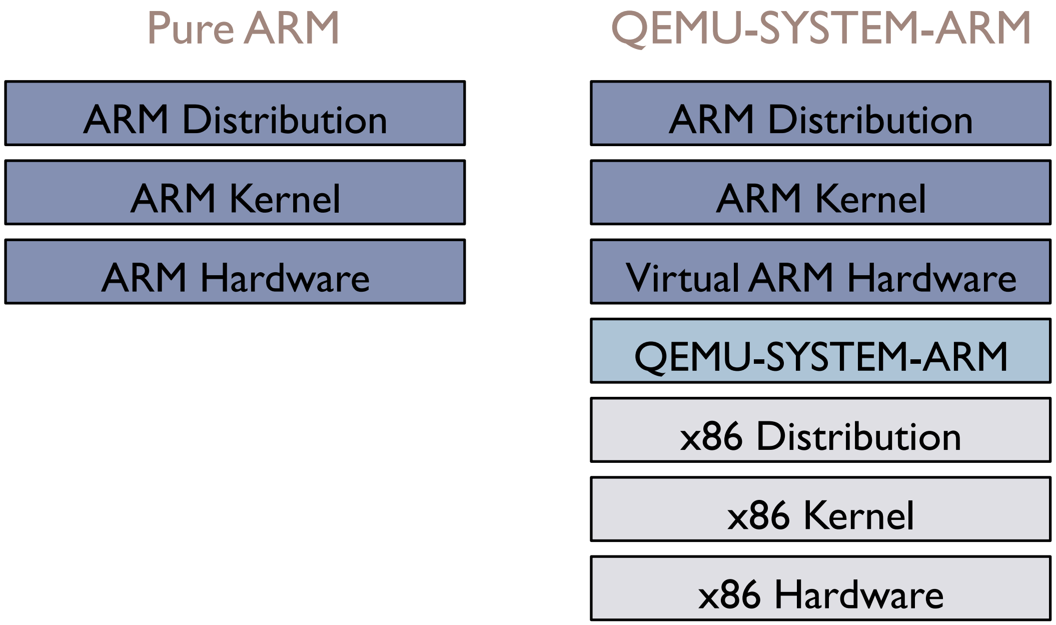
ARMv5EL= EABI, soft FP, little endian, v5, non-thumb

Challenge: Handling the “native” problem

Desktop distributions are not cross-built: you need an ARM-based machine to build an ARM-based distribution

- Option #1: Fundamentally change the build system using something like Scratchbox.
  - We couldn’t find a good way to do this without a lot of source package modifications
- Option #2: Create a build cluster of ARM-based machines...

Options for “native” build machines



Pure ARM

- ARM Distribution
- ARM Kernel
- ARM Hardware

QEMU-SYSTEM-ARM

- ARM Distribution
- ARM Kernel
- Virtual ARM Hardware
- QEMU-SYSTEM-ARM
- x86 Distribution
- x86 Kernel
- x86 Hardware

In 2007 we looked at the time and cost to build a sufficiently fast cluster

2007 cluster: Native ARM build machines

20 home-built 1U ARM boxes

- 600 MHz ARMv5 processors
- 32 hours to compile and run the test suite for gcc-4.1 (one box)
- 4 days to compile the main Ubuntu packages (about 3000)

2008 cluster: Virtual ARM build machines

17 Dell x86 workstations
(34 virtual ARMv6/7 machines)

- Emulated ARMv6 processor (2 on each workstation)
- 25 hours to compile and run test suites for gcc-4.1
- 2 days to compile main Ubuntu packages (about 3000)

About a 25% performance increase and a 60% cost decrease

Outline

- The challenges in building the distributions
  - Compilers, libraries, and toolchains
  - Native machine clusters
- The current state of the distributions
  - What works, what has been patched, what is missing
- How to use a Mojo distribution
  - Sample installation
  - Examples of systems that use the distribution
  - Performance
- Future work

Mojo Releases

			ARM Targets		
Mojo	**Ubuntu**	**Released**	**v5**	**v5+VFP**	**v6+VFP**
Frisky	7.04	March 2008	✔		✔
Grumpy	7.10	June 2008	✔		✔
Hasty	8.04	July 2008	✔	✔	✔
Icy	8.10	Nov. 2008	✔	○	○

- Frisky was our test case
- Grumpy only exists as a bridge to Hasty
- Hasty is in very good shape and has updates
- Icy is very new

Statistics from a sample build

Hasty ARMv5 EL	Main	Contributed
Ubuntu source packages	3114	11188
Ubuntu binary packages	6151	18955
Source packages we modified	55	6
Completely built source packages	2921 (94%)	9591 (86%)

The majority of packages just build without modification

Why didn’t some packages build?

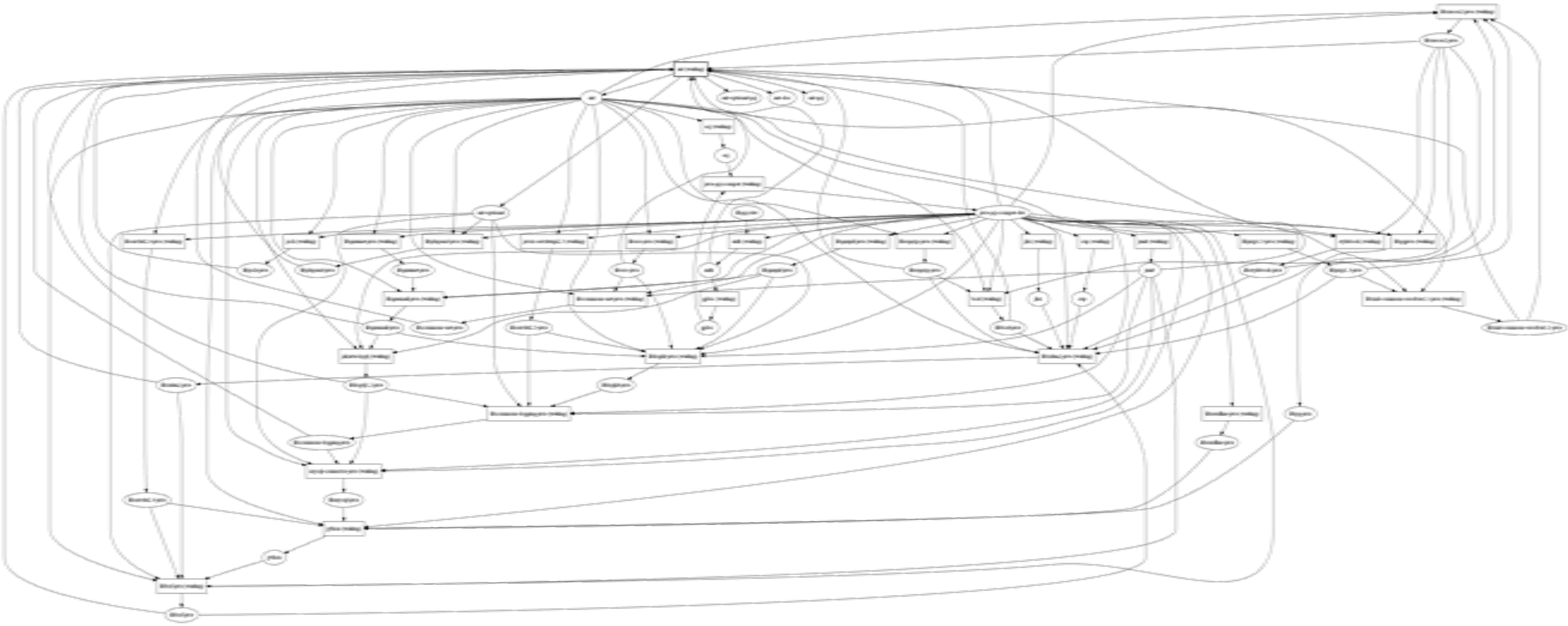
Hasty ARMv5EL	Main	Contributed
Wrong architecture (not ARM)	58 (29%)	247 (15%)
Partially built – some of generated debs are not for ARM (e.g., Linux kernel)	36 (18%)	33 (2%)
Waiting on other packages that failed	81 (41%)	993 (61%)
Failed (e.g., Java)	22 (11%)	364 (22%)

Many packages fail to build completely because they aren’t for ARM

Challenges in package building

- Many Ubuntu packages come in build-dependency loops that must be manually unwound (e.g. KDE, Java)
- Some important packages simply don’t exist for ARM (e.g., Java)
- Some important packages have to be backported from later distributions (e.g. Mono, Fortran compilers)
- Some key packages have to be patched because we’re not officially part of Ubuntu (e.g., dpkg, apt, keyrings)
- Some packages just have errors (e.g., Qt float data type, minor fixes in Python)

A sample dependency loop (Java)



Outline

- The challenges in building the distributions
  - Compilers, libraries, and toolchains
  - Native machine clusters
- The current state of the distributions
  - What works, what has been patched, what is missing
- How to use a Mojo distribution
  - Sample installation
  - Examples of systems that use the distribution
  - Performance
- Future work

Debian Installer with QEMU

- Download installer components

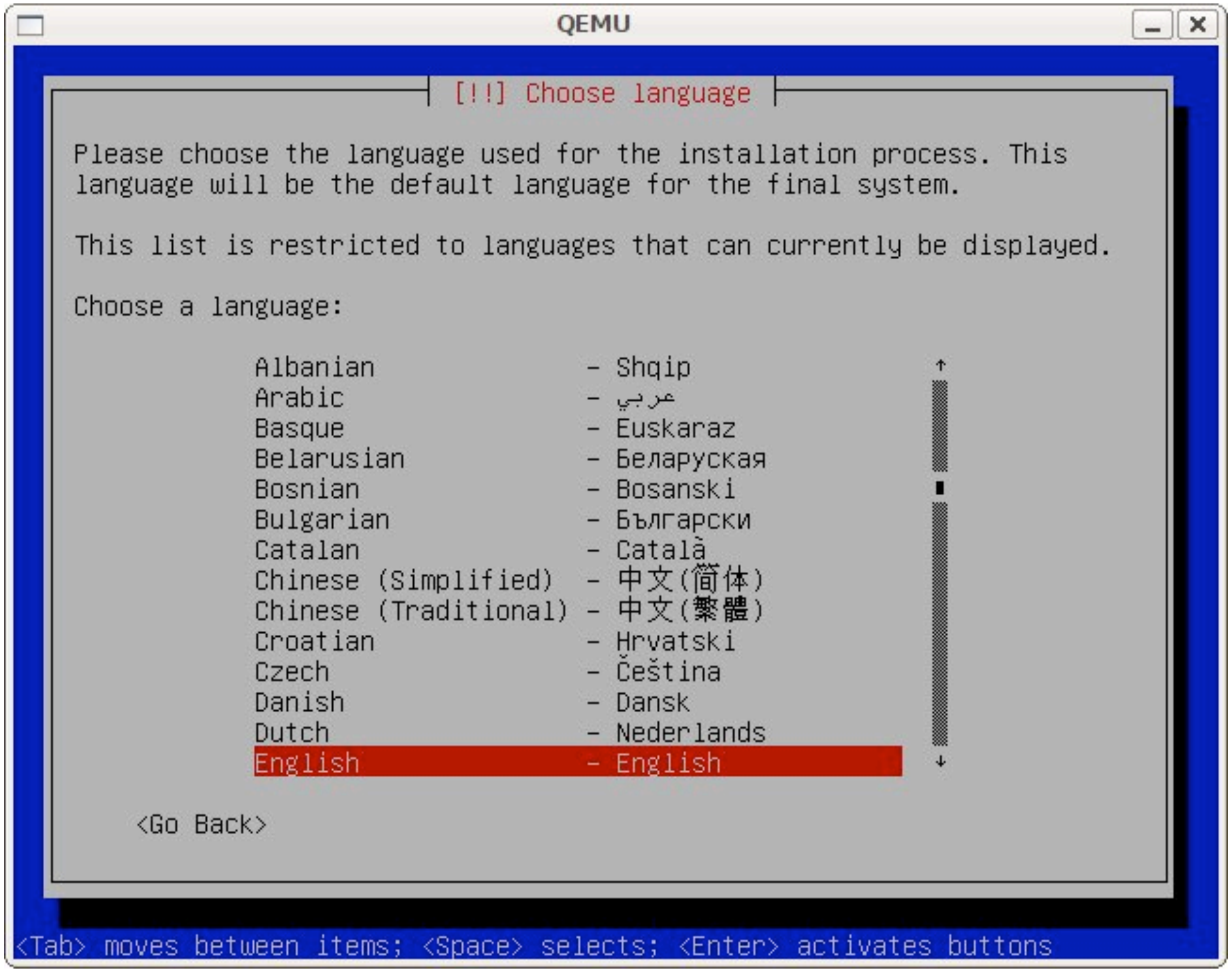
```
$ wget http://repository.handhelds.org/hasty-armv5el/installer-arm/images/versatilepb/ramdisk.gz
$ wget http://repository.handhelds.org/hasty-armv5el/installer-arm/images/versatilepb/vmlinuz-926
```

- Create a QEMU disk image

```
$ qemu-img create –f raw test.img 2G
$ qemu-system-arm –M versatilepb –m 256M –kernel vmlinux-926 \
                    –initrd ramdisk.gz –hda test.img –append “root=/dev/ram”
```

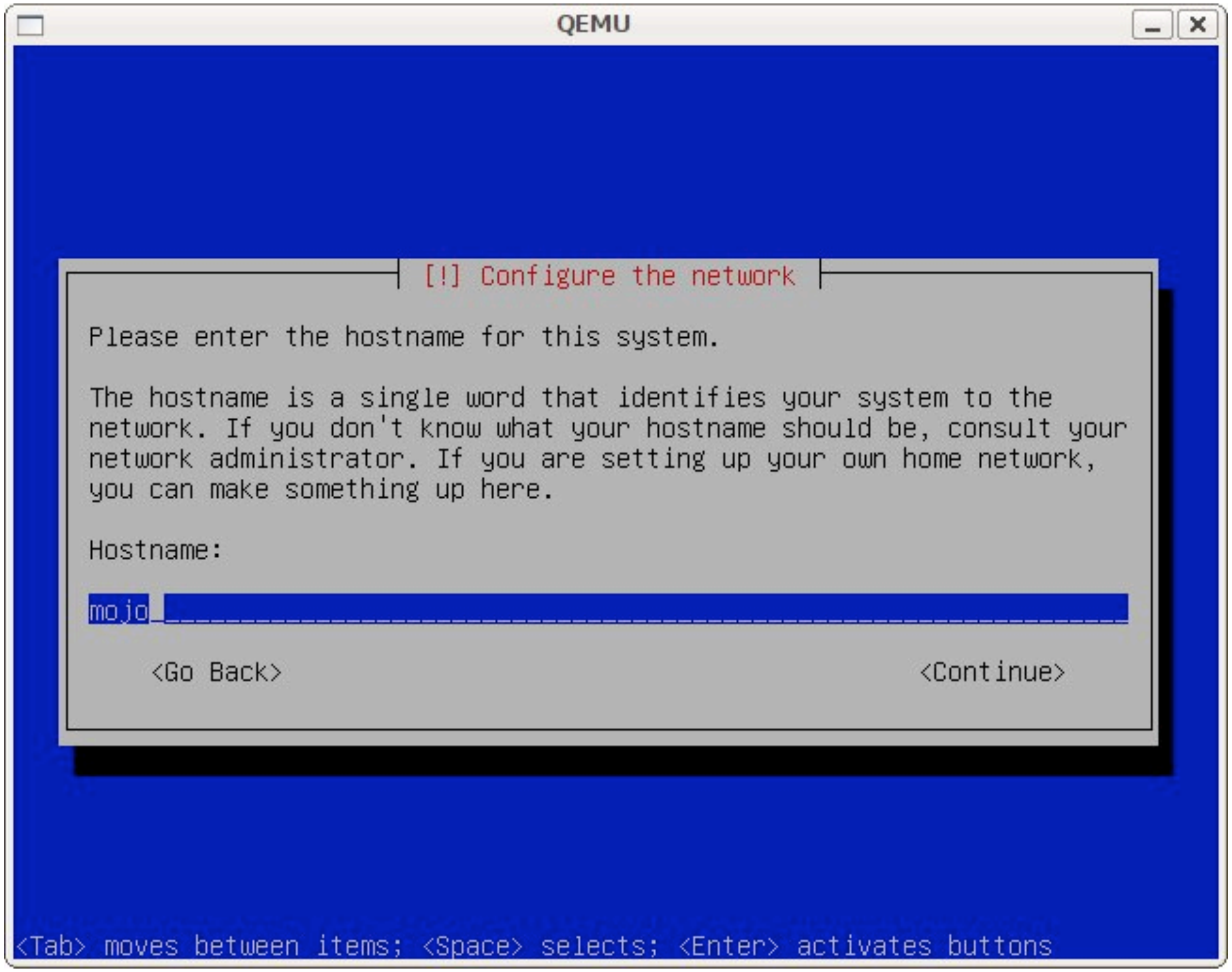
- Run the Debian Installer…

Debian Installer



```
[!!] Choose language

Please choose the language used for the installation process. This
language will be the default language for the final system.

This list is restricted to languages that can currently be displayed.

Choose a language:

            Albanian                 - Shqip
            Arabic                   - عربي
            Basque                   - Euskaraz
            Belarusian               - Беларуская
            Bosnian                  - Bosanski
            Bulgarian                - Български
            Catalan                  - Català
            Chinese (Simplified)     - 中文(简体)
            Chinese (Traditional)    - 中文(繁體)
            Croatian                 - Hrvatski
            Czech                    - Čeština
            Danish                   - Dansk
            Dutch                    - Nederlands
            English                  - English

    <Go Back>

<Tab> moves between items; <Space> selects; <Enter> activates buttons
```

Debian Installer



[!] Configure the network

Please enter the hostname for this system.

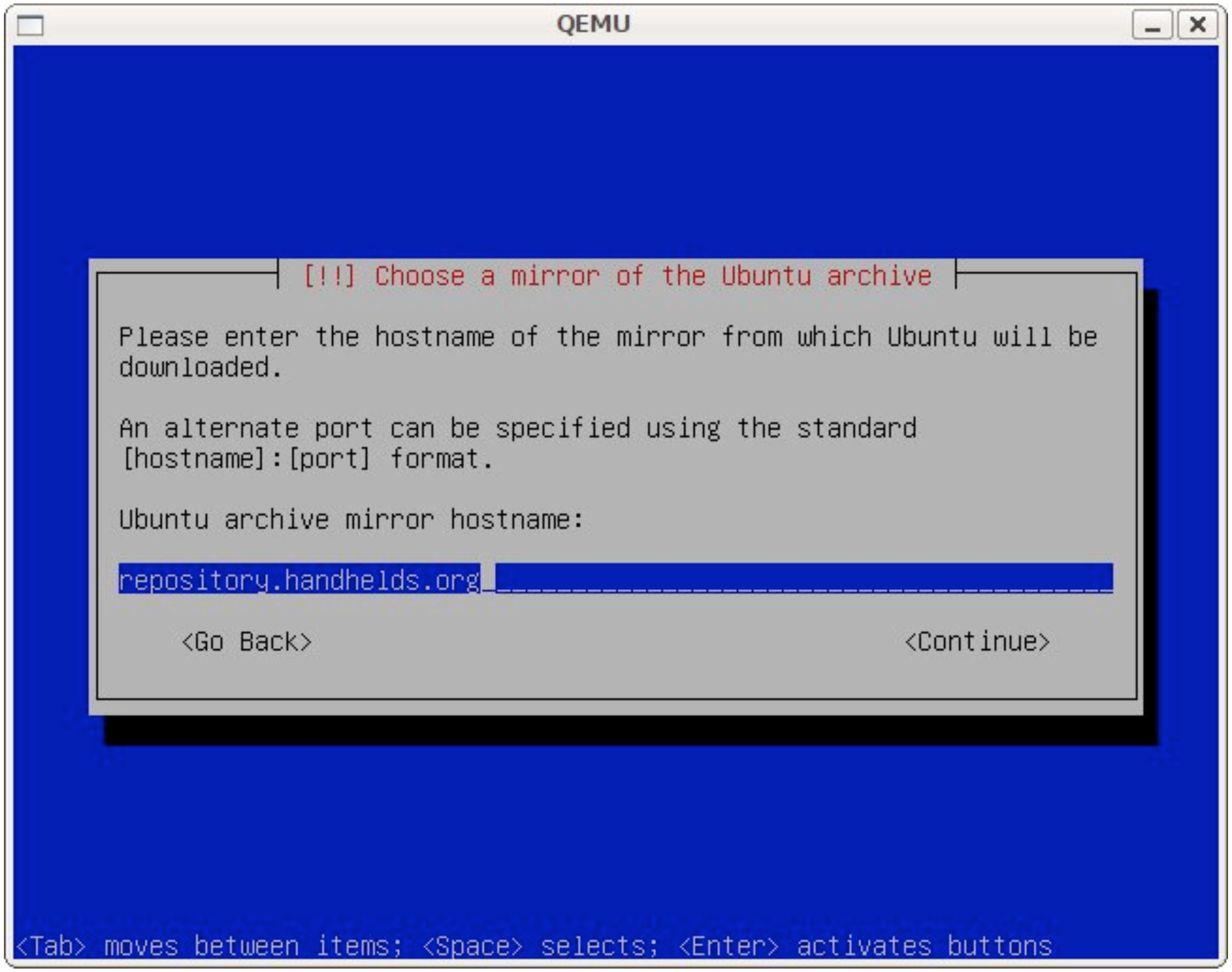
The hostname is a single word that identifies your system to the network. If you don't know what your hostname should be, consult your network administrator. If you are setting up your own home network, you can make something up here.

Hostname:

mojo

<Tab> moves between items; <Space> selects; <Enter> activates buttons

Debian Installer



[!!] Choose a mirror of the Ubuntu archive

Please enter the hostname of the mirror from which Ubuntu will be downloaded.

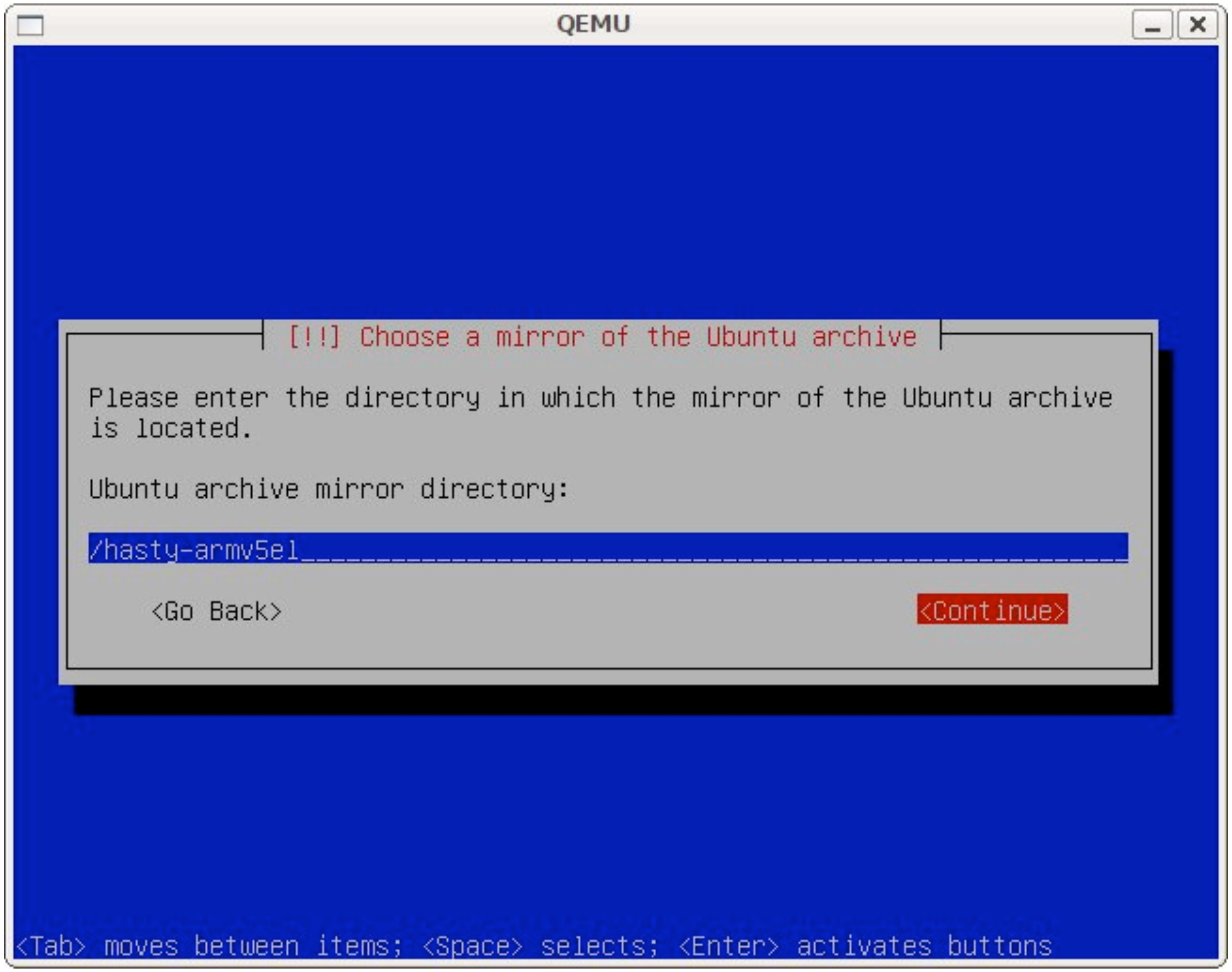
An alternate port can be specified using the standard [hostname]:[port] format.

Ubuntu archive mirror hostname:

repository.handhelds.org

<Tab> moves between items; <Space> selects; <Enter> activates buttons

Debian Installer



[!!] Choose a mirror of the Ubuntu archive

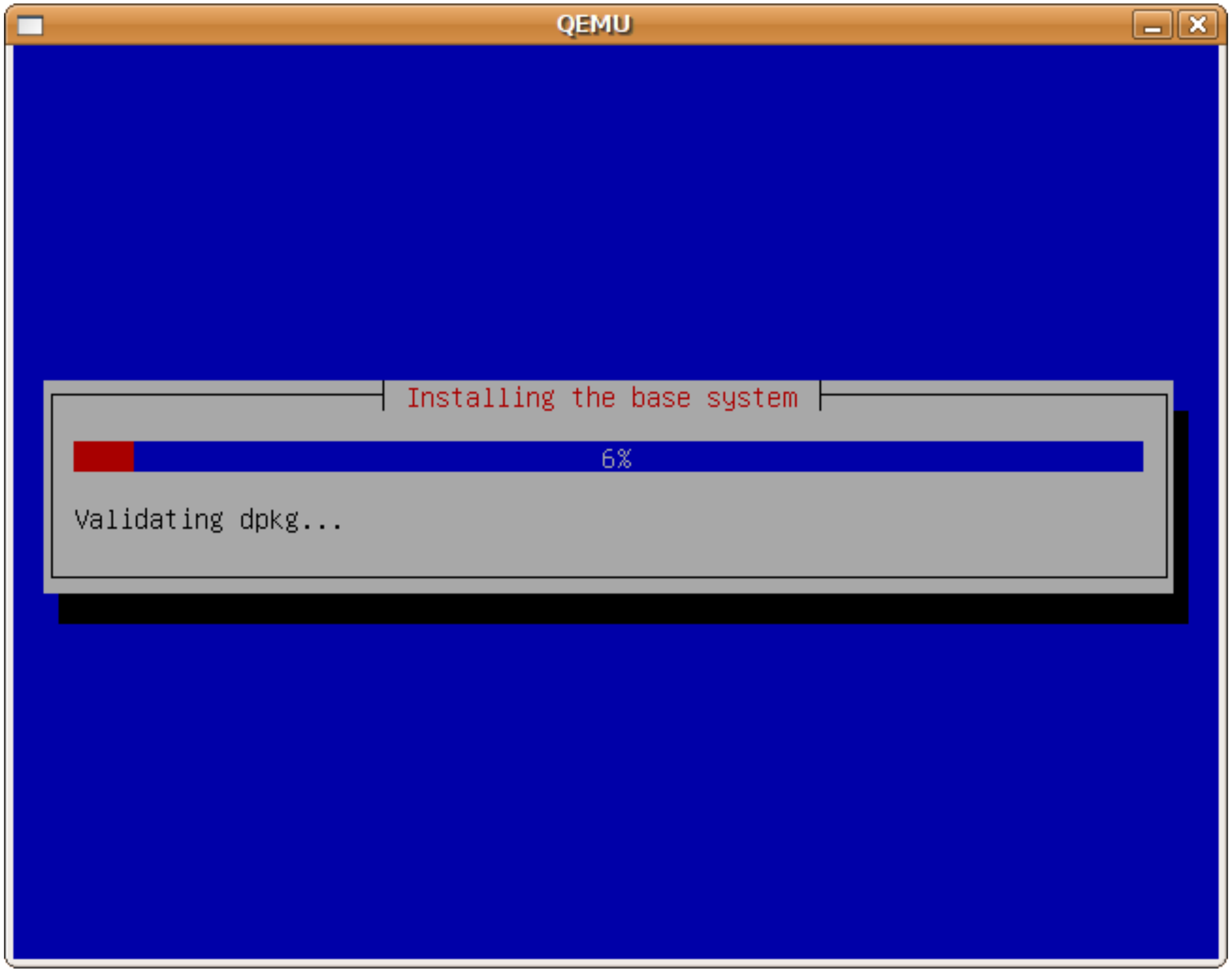
Please enter the directory in which the mirror of the Ubuntu archive is located.

Ubuntu archive mirror directory:

/hasty-armv5el

<Tab> moves between items; <Space> selects; <Enter> activates buttons

Debian Installer

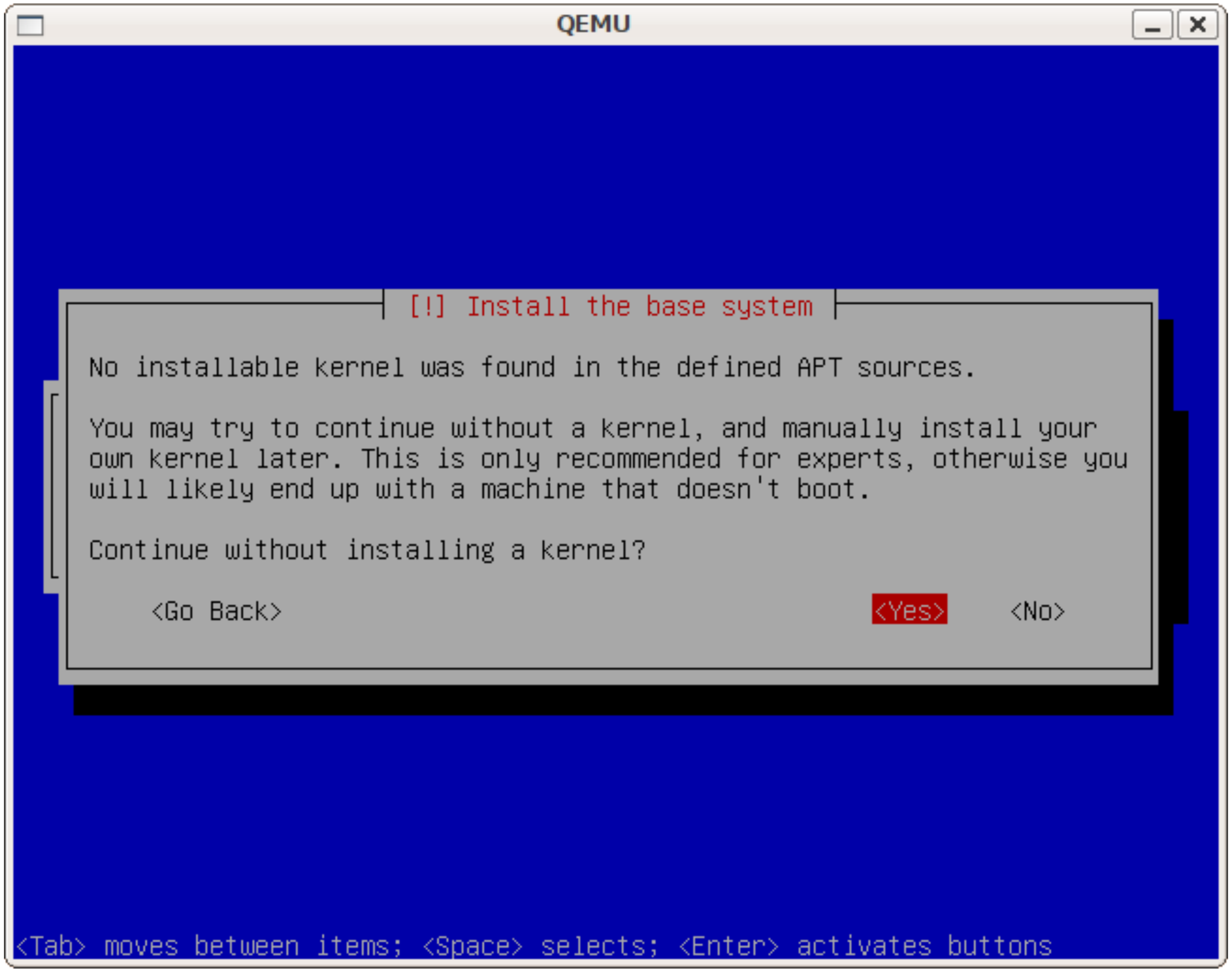


Installing the base system

6%

Validating dpkg...

Debian Installer



[!] Install the base system

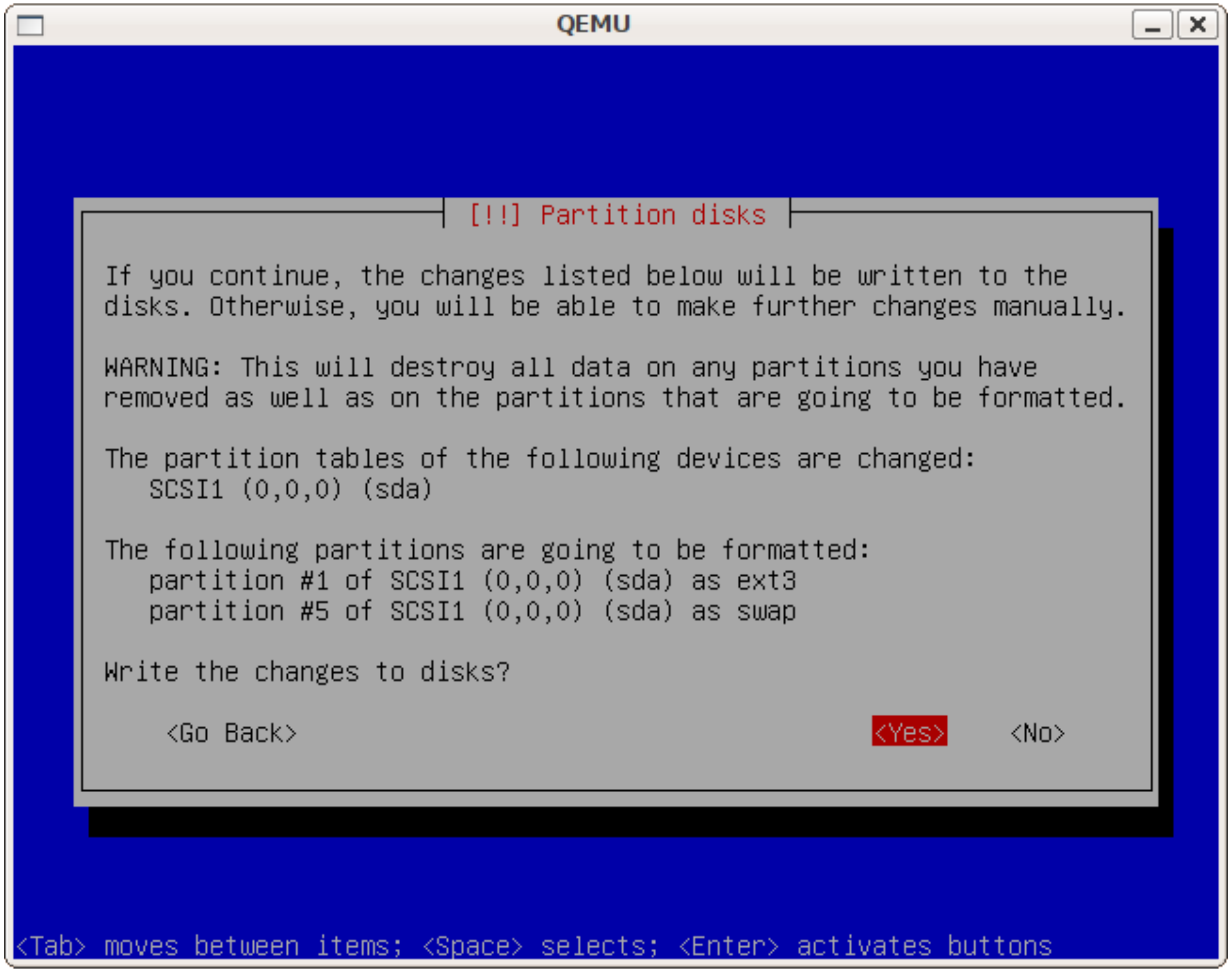
No installable kernel was found in the defined APT sources.

You may try to continue without a kernel, and manually install your own kernel later. This is only recommended for experts, otherwise you will likely end up with a machine that doesn't boot.

Continue without installing a kernel?

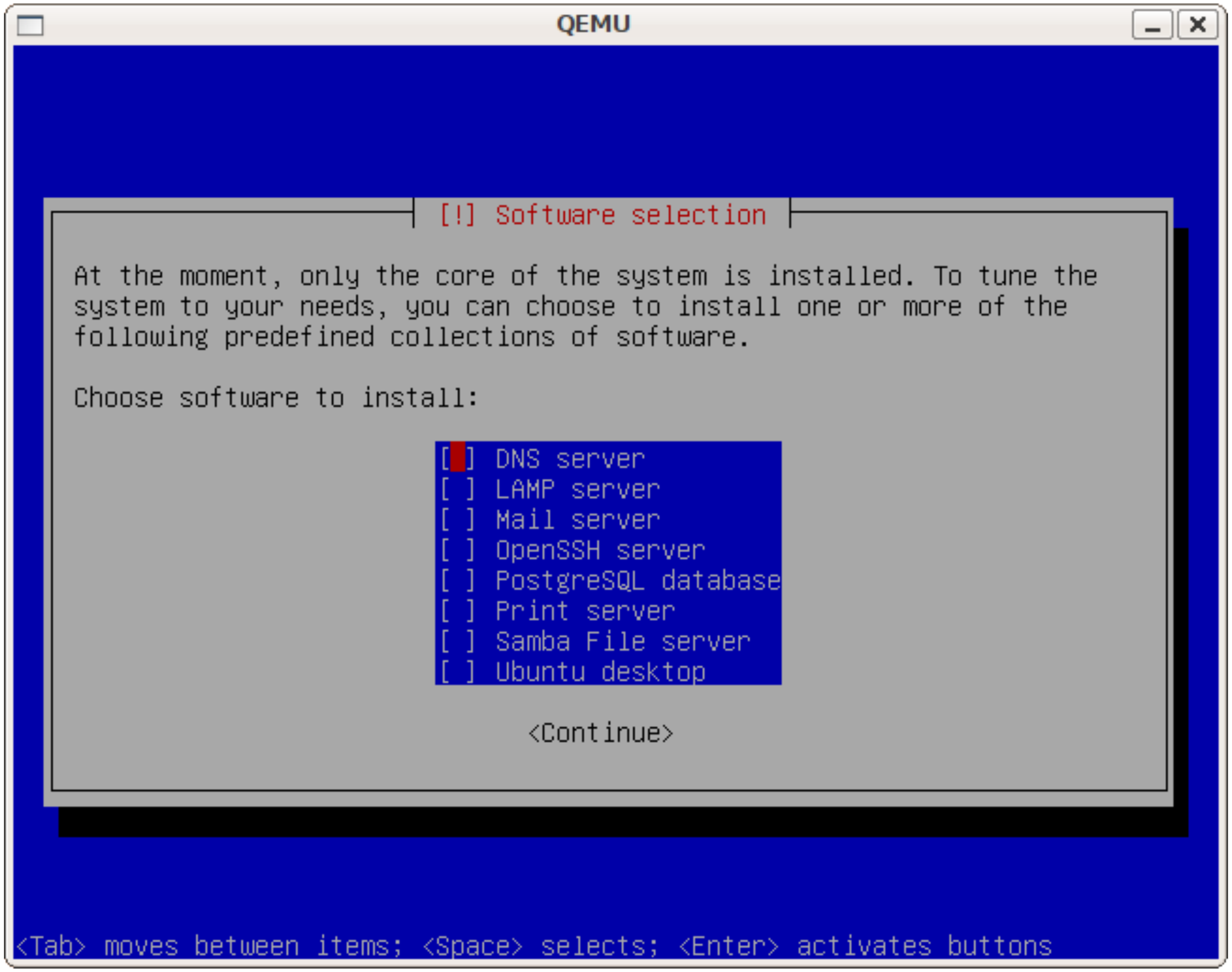
<Tab> moves between items; <Space> selects; <Enter> activates buttons

Debian Installer



```
[!!] Partition disks

If you continue, the changes listed below will be written to the
disks. Otherwise, you will be able to make further changes manually.

WARNING: This will destroy all data on any partitions you have
removed as well as on the partitions that are going to be formatted.

The partition tables of the following devices are changed:
   SCSI1 (0,0,0) (sda)

The following partitions are going to be formatted:
   partition #1 of SCSI1 (0,0,0) (sda) as ext3
   partition #5 of SCSI1 (0,0,0) (sda) as swap

Write the changes to disks?

     <Go Back>                                       <Yes>     <No>

<Tab> moves between items; <Space> selects; <Enter> activates buttons
```

Debian Installer



[!] Software selection

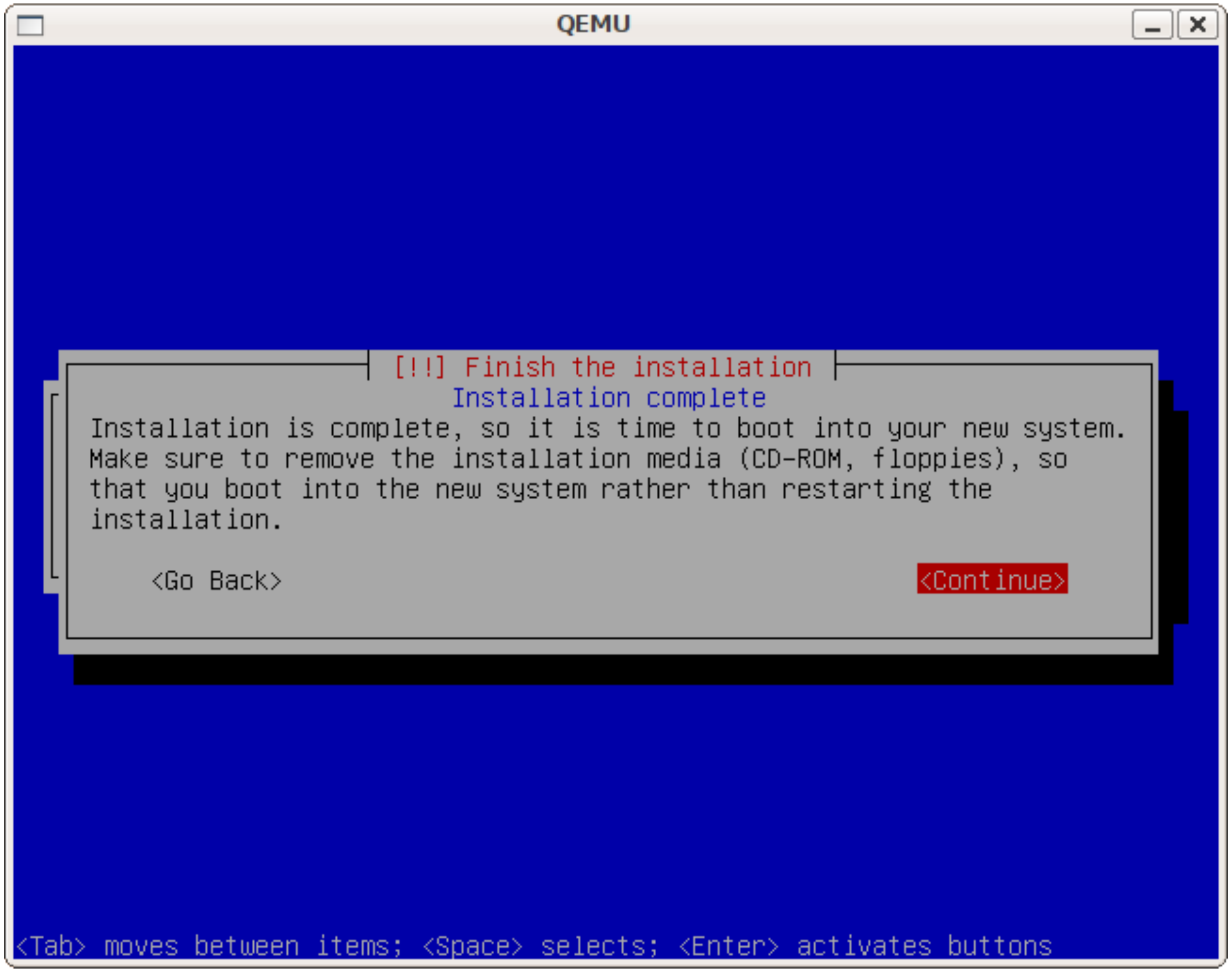
At the moment, only the core of the system is installed. To tune the system to your needs, you can choose to install one or more of the following predefined collections of software.

Choose software to install:

```
[ ] DNS server
[ ] LAMP server
[ ] Mail server
[ ] OpenSSH server
[ ] PostgreSQL database
[ ] Print server
[ ] Samba File server
[ ] Ubuntu desktop
```

<Tab> moves between items; <Space> selects; <Enter> activates buttons

Debian Installer



[!!] Finish the installation

Installation complete

Installation is complete, so it is time to boot into your new system. Make sure to remove the installation media (CD-ROM, floppies), so that you boot into the new system rather than restarting the installation.

<Tab> moves between items; <Space> selects; <Enter> activates buttons

Moving beyond the installer

- Once the installer has finished, you can boot the image in QEMU with:

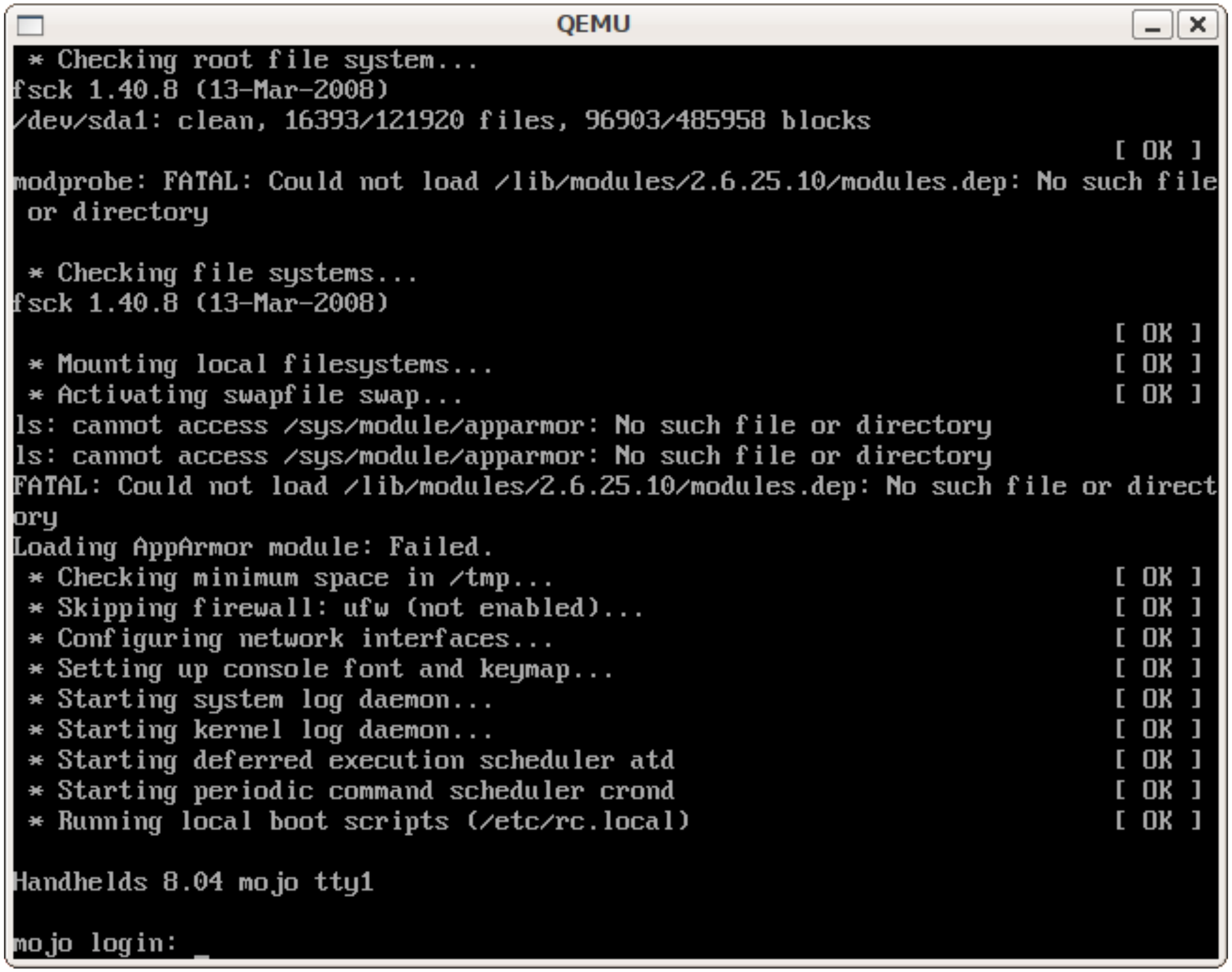
```
$ qemu-system-arm –M versatilepb –m 256M –kernel vmlinuz-926 –hda test.img \
                              –append “root=/dev/sda1”
```

- If you’d like a graphical environment, try:

```
$ apt-get install xorg xfce4 gdm
# Edit /etc/X11/xorg.conf to include Driver “fbdev”
```

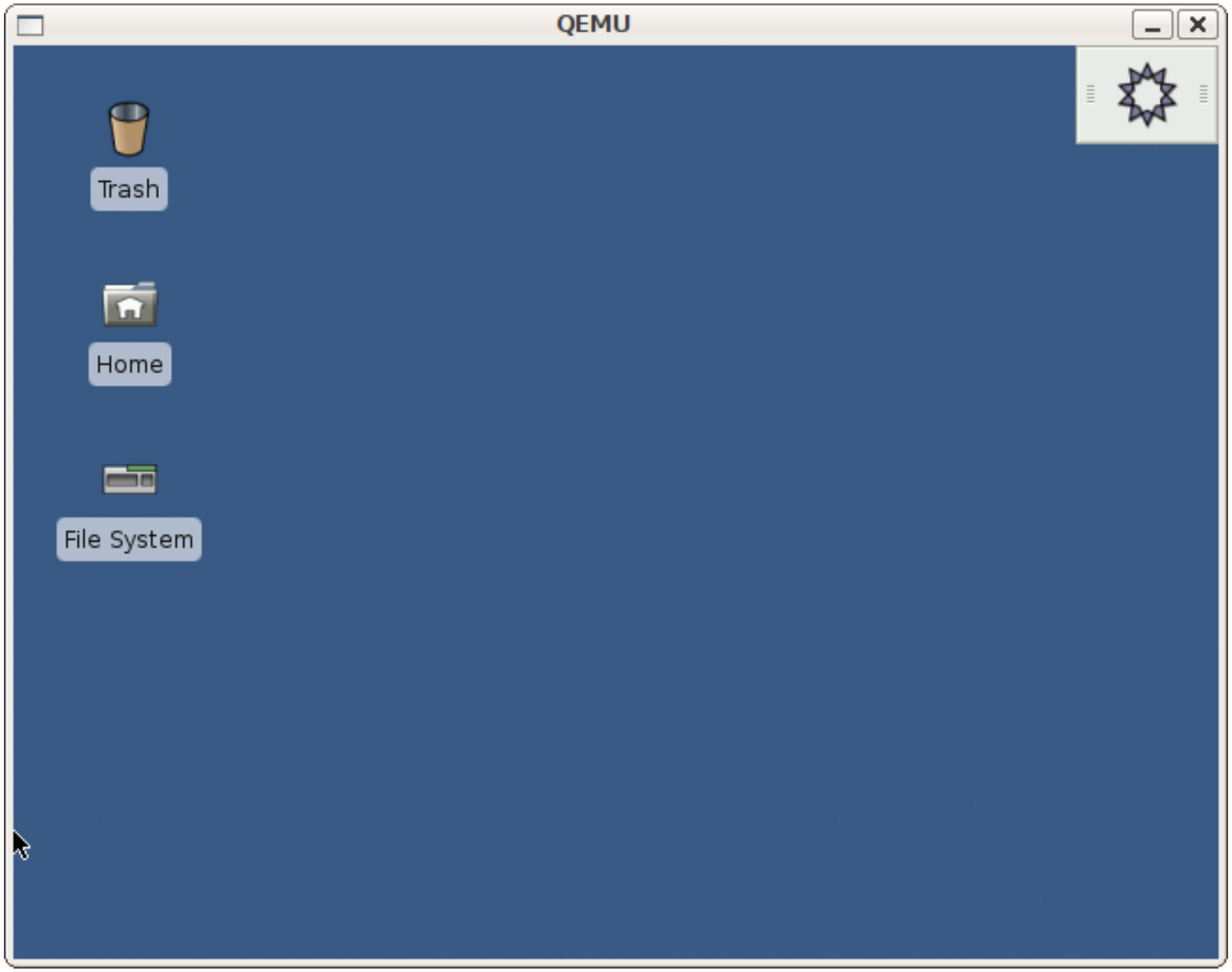
Instructions are on the website and the Mojo wiki

Running system



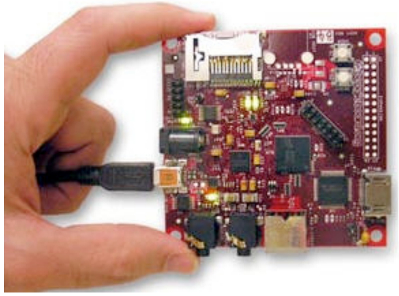
```
 * Checking root file system...
fsck 1.40.8 (13-Mar-2008)
/dev/sda1: clean, 16393/121920 files, 96903/485958 blocks
                                                                        [ OK ]
modprobe: FATAL: Could not load /lib/modules/2.6.25.10/modules.dep: No such file
 or directory

 * Checking file systems...
fsck 1.40.8 (13-Mar-2008)
                                                                        [ OK ]
 * Mounting local filesystems...                                        [ OK ]
 * Activating swapfile swap...                                          [ OK ]
ls: cannot access /sys/module/apparmor: No such file or directory
ls: cannot access /sys/module/apparmor: No such file or directory
FATAL: Could not load /lib/modules/2.6.25.10/modules.dep: No such file or direct
ory
Loading AppArmor module: Failed.
 * Checking minimum space in /tmp...                                    [ OK ]
 * Skipping firewall: ufw (not enabled)...                              [ OK ]
 * Configuring network interfaces...                                    [ OK ]
 * Setting up console font and keymap...                                [ OK ]
 * Starting system log daemon...                                        [ OK ]
 * Starting kernel log daemon...                                        [ OK ]
 * Starting deferred execution scheduler atd                            [ OK ]
 * Starting periodic command scheduler crond                            [ OK ]
 * Running local boot scripts (/etc/rc.local)                           [ OK ]

Handhelds 8.04 mojo tty1

mojo login: _
```

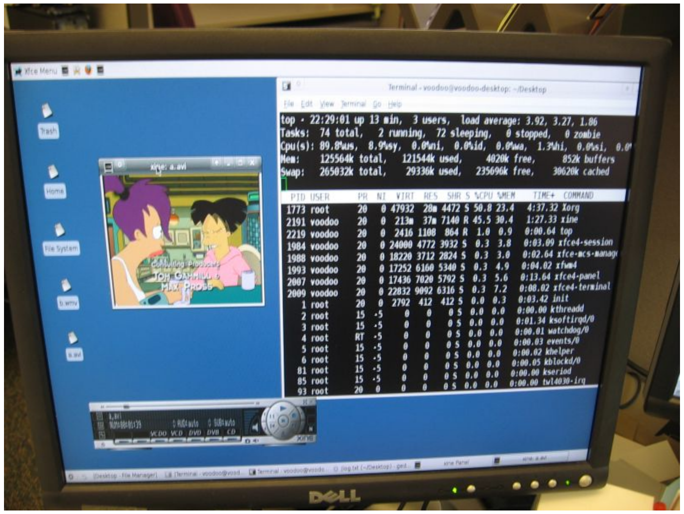
After running “startx”



Examples of what you can do

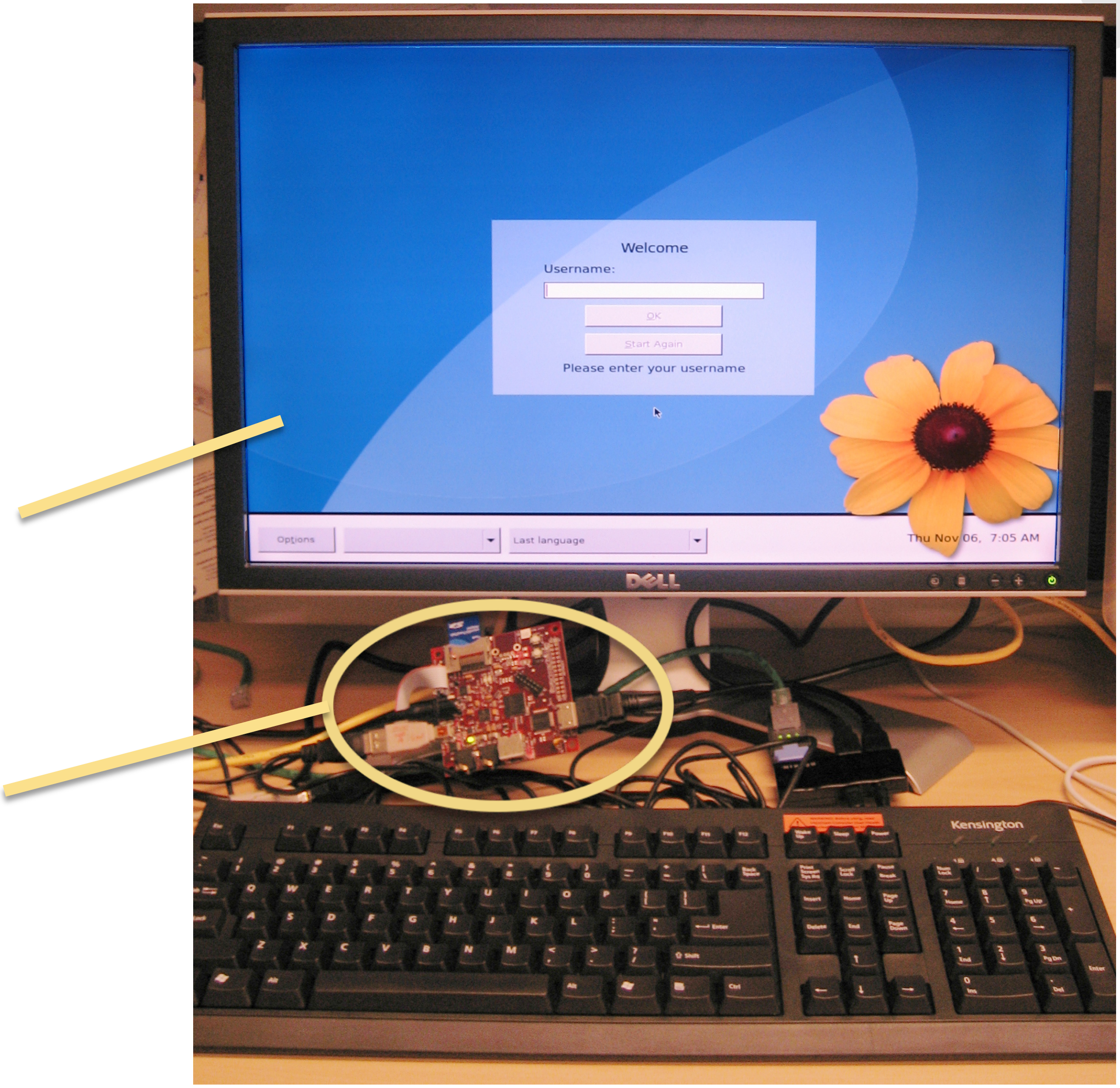
- Robert Nelson has Beagle Board instructions
- Cortez has been working on the Sharp Zaurus
- Rabeeh Khoury (Marvell) has good stuff for the Marvell 78100 board (wicked fast ARM…)




Terminal - voodoo@voodoo-desktop: ~/Desktop
File Edit View Terminal Go Help
DELL

```
top - 22:29:01 up 13 min,  3 users,  load average: 3.92, 3.27, 1.86
Tasks:  74 total,   2 running,  72 sleeping,   0 stopped,   0 zombie
Cpu(s): 89.8%us,  8.9%sy,  0.0%ni,  0.0%id,  0.0%wa,  1.3%hi,  0.0%si,  0.0
Mem:    125564k total,   121544k used,     4020k free,      852k buffers
Swap:   265032k total,    29336k used,   235696k free,    30620k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1773 root      20   0 47932  28m 4472 S 50.8 23.4   4:37.32 Xorg
 2191 voodoo    20   0  213m  37m 7140 R 45.5 30.4   1:27.33 xine
 2219 voodoo    20   0  2416 1108  864 R  1.0  0.9   0:00.64 top
 1984 voodoo    20   0 24000 4772 3932 S  0.3  3.8   0:03.09 xfce4-session
 1988 voodoo    20   0 18220 3712 2824 S  0.3  3.0   0:02.64 xfce-mcs-manage
 1993 voodoo    20   0 17252 6160 5340 S  0.3  4.9   0:04.02 xfwm4
 2007 voodoo    20   0 17436 7020 5792 S  0.3  5.6   0:13.64 xfce4-panel
 2009 voodoo    20   0 22832 9092 6316 S  0.3  7.2   0:08.02 xfce4-terminal
    1 root      20   0  2792  412  412 S  0.0  0.3   0:03.42 init
    2 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      15  -5     0    0    0 S  0.0  0.0   0:01.34 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.01 watchdog/0
    5 root      15  -5     0    0    0 S  0.0  0.0   0:00.03 events/0
    6 root      15  -5     0    0    0 S  0.0  0.0   0:00.02 khelper
   81 root      15  -5     0    0    0 S  0.0  0.0   0:00.05 kblockd/0
   85 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
   93 root      20   0     0    0    0 S  0.0  0.0   0:00.00 twl4030-irq
```

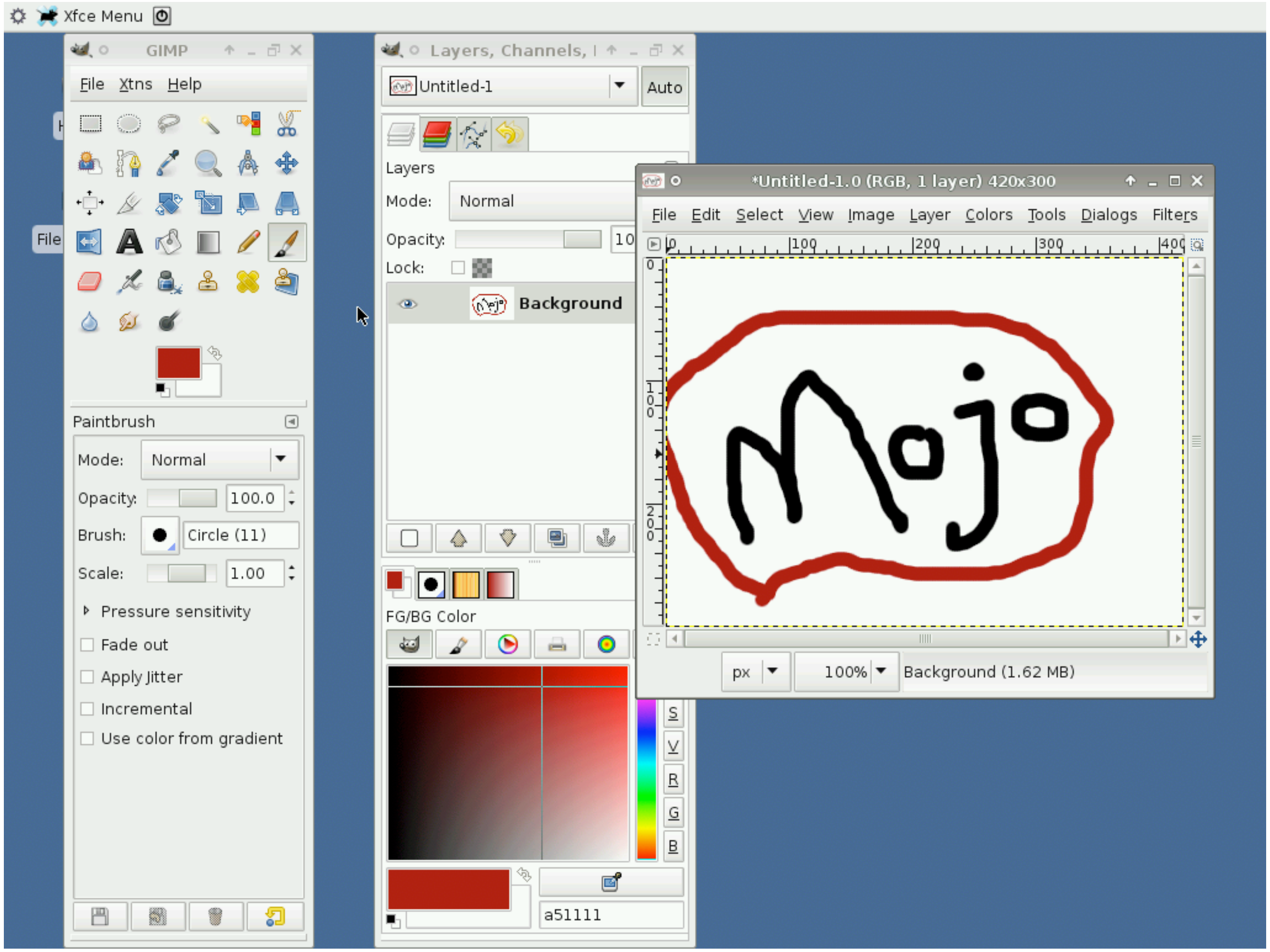
Testing Hasty ARMv5EL & v6EL-VFP



XFCE4 + gdm

Beagleboard

Gimp on a BeagleBoard (Hasty v5)



Mode: Normal

Opacity: 100.0

Brush: ● Circle (11)

Scale: 1.00

- Pressure sensitivity

□ Fade out

□ Apply Jitter

□ Incremental

□ Use color from gradient

Mode: Normal

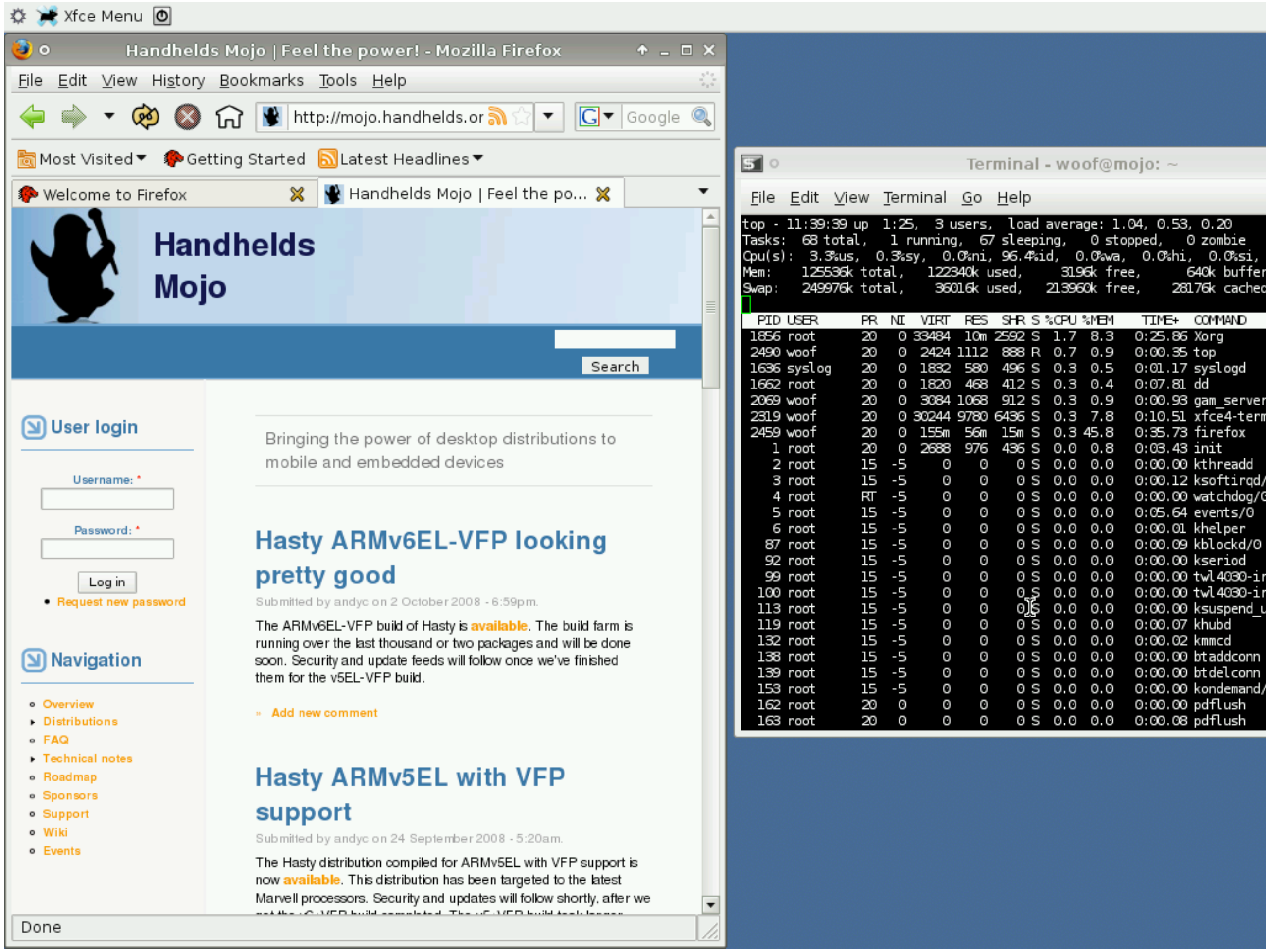
Opacity: 10

Lock: □

px 100% Background (1.62 MB)

a51111

Firefox on BeagleBoard (Hasty v5)


Xfce Menu
Handhelds Mojo | Feel the power! - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://mojo.handhelds.or
Google
Most Visited
Getting Started
Latest Headlines
Welcome to Firefox
Handhelds Mojo | Feel the po...
Handhelds
Mojo
Search
Done
Terminal - woof@mojo: ~
File Edit View Terminal Go Help

User login

Username: *

Password: *

Log in

- Request new password

Navigation

- Overview
- Distributions
- FAQ
- Technical notes
- Roadmap
- Sponsors
- Support
- Wiki
- Events

Bringing the power of desktop distributions to mobile and embedded devices

Hasty ARMv6EL-VFP looking pretty good

Submitted by andyc on 2 October 2008 - 6:59pm.

The ARMv6EL-VFP build of Hasty is available. The build farm is running over the last thousand or two packages and will be done soon. Security and update feeds will follow once we've finished them for the v5EL-VFP build.

» Add new comment

Hasty ARMv5EL with VFP support

Submitted by andyc on 24 September 2008 - 5:20am.

The Hasty distribution compiled for ARMv5EL with VFP support is now available. This distribution has been targeted to the latest Marvell processors. Security and updates will follow shortly, after we

```
top - 11:39:39 up  1:25,  3 users,  load average: 1.04, 0.53, 0.20
Tasks:  68 total,   1 running,  67 sleeping,   0 stopped,   0 zombie
Cpu(s):  3.3%us,  0.3%sy,  0.0%ni, 96.4%id,  0.0%wa,  0.0%hi,  0.0%si,
Mem:    125536k total,   122340k used,     3196k free,      640k buffer
Swap:   249976k total,    36016k used,   213960k free,    28176k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1856 root      20   0 33484  10m 2592 S  1.7  8.3   0:25.86 Xorg
 2490 woof      20   0  2424 1112  888 R  0.7  0.9   0:00.35 top
 1636 syslog    20   0  1832  580  496 S  0.3  0.5   0:01.17 syslogd
 1662 root      20   0  1820  468  412 S  0.3  0.4   0:07.81 dd
 2069 woof      20   0  3084 1068  912 S  0.3  0.9   0:00.93 gam_server
 2319 woof      20   0 30244 9780 6436 S  0.3  7.8   0:10.51 xfce4-term
 2459 woof      20   0  155m  56m  15m S  0.3 45.8   0:35.73 firefox
    1 root      20   0  2688  976  436 S  0.0  0.8   0:03.43 init
    2 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      15  -5     0    0    0 S  0.0  0.0   0:00.12 ksoftirqd/
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      15  -5     0    0    0 S  0.0  0.0   0:05.64 events/0
    6 root      15  -5     0    0    0 S  0.0  0.0   0:00.01 khelper
   87 root      15  -5     0    0    0 S  0.0  0.0   0:00.09 kblockd/0
   92 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
   99 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 twl4030-ir
  100 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 twl4030-ir
  113 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 ksuspend_u
  119 root      15  -5     0    0    0 S  0.0  0.0   0:00.07 khubd
  132 root      15  -5     0    0    0 S  0.0  0.0   0:00.02 kmmcd
  138 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 btaddconn
  139 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 btdelconn
  153 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kondemand/
  162 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  163 root      20   0     0    0    0 S  0.0  0.0   0:00.08 pdflush
```

Does architecture matter?

- A quick performance test using Cairo to draw falling, spinning PNG and SVG files
- Tested on a TI BeagleBoard, we saw a 15-20% speedup from the Hasty ARMv5EL distribution to the Hasty ARMv6EL+VFP

Future work

- Building the Icy (8.10) release
- Submitting patches back to Debian and Ubuntu
- Considering adding a new architecture or two
- Considering building Debian

…and *using* these distributions, of course…

http://mojo.handhelds.org