Casey DeLorme's Comprehensive Guide to Xen Linux

Missing: VNC Console Configuration, Screenshots for ATI Driver Installation, Hand Drawn LVM Diagram Examples

Table of Contents

Forward

What is this Guide?

This is an all-inclusive walkthrough from preparing materials for installation to successfully installing your graphics card drivers in a virtual machine.

Besides being very long and detailed it was also specifically written for my hardware. Factors such as change in hardware, chosen OS (Dom0), kernel version & flags, as well as configuration and software package versions can all affect your success.

This guide may not work for nVidia graphics cards, currently they are lacking a lot of support. Patches exist, but success is rare with modern Windows installations.

Why a Single Guide?

As mentioned, so many factors can create different results with Xen, and my experience with partial guides has been just that. I wanted to create a single guide and would rather it remain as such.

Why VGA Passthrough?

The simple and obvious answer is Gaming, but there is a bit more detail behind it than that. Games are designed for Windows, end of story. However, to play those games Windows needs a powerful video card to make it work. With hardware virtualization we can now play HD video in a virtual machine, but we still don't have the power needed to run modern video games.

My experience has been with VMWare Server 2 &. Workstation 8, for just over 3 years I used Windows Server 2008 R2 as a host operating system so I could still play games, and my VM's consisted of a development debian web server and a PFSense router that handled my home networks traffic.

Honestly, Core i7 processors are absurdly powerful, and I was aiming for efficiency. Instead of forking out the cost of creating two new smaller machines for PFSense and a web server, I combined it into one, and still barely touched the processing power available. However, it's biggest flaw was Windows. The biggest target for viruses, so I had to make sure I had a backup image available, but the most annoying thing was actually Windows Updates. Consider a decision where you can take down your network for a few minutes once or twice a week, or expose known security holes in your system?

My answer to the larger question, why VGA Passthrough, is to resolve my Windows dependency using modern technology. With IOMMU I can now pass my graphics card to a virtual machine getting near-native performance and keep the core of my system running on a "real" server platform.

Why EFI?

Consumer motherboards finally have it, and UEFI BIOS is amazing. EFI Bootloaded systems allow you to use newer more modern partitioning schemes and a better boot architecture. MBR BIOS Bootloaders are over 20 years old, and while that's great for stability of older systems I want to move forward modern superior technology.

A Humble Request

This guides verbosity may bore experienced Linux users, but I want to ask that all users who try this guide add the following:

- Any additional steps or additions to steps you feel are unclear
- Your hardware
- Deviations from the guide
- Whether you were successful
- Bugs You Encountered

I want this guide to serve novice Linux users, and I also want more documentation for Xen to be available as it is lacking it in many areas. Since Xen is open-source one of the best resources for new and improved documentation is its users. As such I humbly request that you do as listed, and make your experiences known.

Requirements:

My Hardware

It's fine if you intend to use different hardware, just heed my warning you will probably encounter new situations, which may or may not result on success.

Here is my basic list of equipment as relates to this guide:

Motherboard:	ASRock Z68 Extreme4 Gen3
CPU:	Intel Core i2600
RAM:	12GB 1333Mhz Corsair XMS (2x2G + 2x4G)
Boot Disk:	240GB OCZ Vertex 3
GPU:	ATI Radeon HD 6870
LAN:	Onboard Broadcom BCM57781 & PCIe EXPI9301CTBLK

Various USB & Input Devices:

Dom0 USB 3.0: Logitech K320 & Nano Receiver, Logitech M305 Wireless Laser Mouse & Unifying Receiver.

HVM USB 2.0: ASUS BT211 BlueTooth Dongle/Adapter, Apple BT Keyboard & Mouse, Logitech C910 HD WebCam, Happauge HD PVR, Bamboo Touch Walcom Tablet. XBox 360 Wireless Dongle/Adapter.

Chosen OS & Required Software Packages

I used Debian Wheezy (testing). We are playing with things that are clearly in development, VGA Passthrough is no perfect science, and at the time of writing Wheezy will be the new "stable" OS in just a few months. My experiences with Wheezy have so far been fantastic.

On Debian Wheezy to compile both a custom Linux Kernel and Xen 4.2, I installed the following packages:

- grub-efi-amd64
- sudo
- ssh
- bridge-utils
- parted
- ntfsprogs
- bzip2
- build-essential
- libncurses-dev
- kernel-package
- fakeroot
- python-dev
- uuid-dev

- libglib2.0-dev
- libyajl-dev
- bcc
- gcc-multilib
- iasl
- libpci-dev
- mercurial

Total space required on the OS disk should be around 217 Megabytes.

I did not separate the packages between Kernel and Xen, since some are shared, and my guide is written for both not one.

Results may vary if you use a different version of Debian, a later revision of Xen 4.2, or a different Dom0 operating system. My best advice is to check the equivalent packages on your operating system and give it a spin.

Materials/Resources

- Debian Wheezy (Testing) Install CD
- UEFI Bootable Ubuntu Live CD (The latest Ubuntu Live CD's should be by default)
- Onboard graphics or a second GPU for Dom0

My guide features a Pure EFI Installation, to make that happen the easiest solution I have found is to use an Ubuntu Live CD to access grub for emergency manual boot procedures. Even if you aren't doing an EFI bootloader, you may find this to be useful if not fascinating.

Without onboard or a secondary graphics card for Dom0, you won't have a way of interfacing with the control system unless you have a second machine and can do everything from SSH. I would consider that risky, and would advise you to have a graphics unit dedicated to Dom0.

User Requirements

While written for novice Linux users, my guide does require the following of its readers:

- Familiar with CLI (Command Line Interface) and comfortable using Terminal
- Are not afraid of data loss (have backed up our files elsewhere)
- Have an Onboard or Secondary GPU for Dom0

Unless you want to use libvirt tools, all Xen commands and maintenance is handled via CLI.

If you don't have a dedicated graphics unit for Dom0 you won't be able to manage Xen and launch your Windows HVM. You could use ssh, but I would not rely on that alone.

Recommendations to make the process easier:

- Familiarity with "vi"
- A second machine for SSH and VNC
- Understand Drive Partitioning

While "vi" is pretty old-school, not knowing how to use it means you have to apply a workaround. Installing a GUI and using a GUI text editor, or editing files over SFTP from another machine with a GUI text editor are examples, but both require significantly more resources and time. Take my advice, learn vi, it will make you a better Linux user. Most of the editing we do is fairly limited, you don't have to be an expert and know all the commands.

If you do not have a second machine to work from, you will need to install the Linux GUI so you can install Windows over VNC.

I have included follow-up sections on both SSH and LVM Partitioning to help give a basic overview of each.

Finally, if you do not plan on using EFI, then be sure you skip all related steps and configuration changes.

Known Hardware Issues

The following issues are in relation to my hardware. If you are using different hardware feel free to skip this section, but if you have hardware issues of your own please add to this section.

Marvell SATA Controller is not Intel IOMMU Compliant, and will throw many boot-delaying errors on Linux. My experience is that the device simply does not function at all with my configuration, I turned it off from UEFI BIOS to avoid error messages.

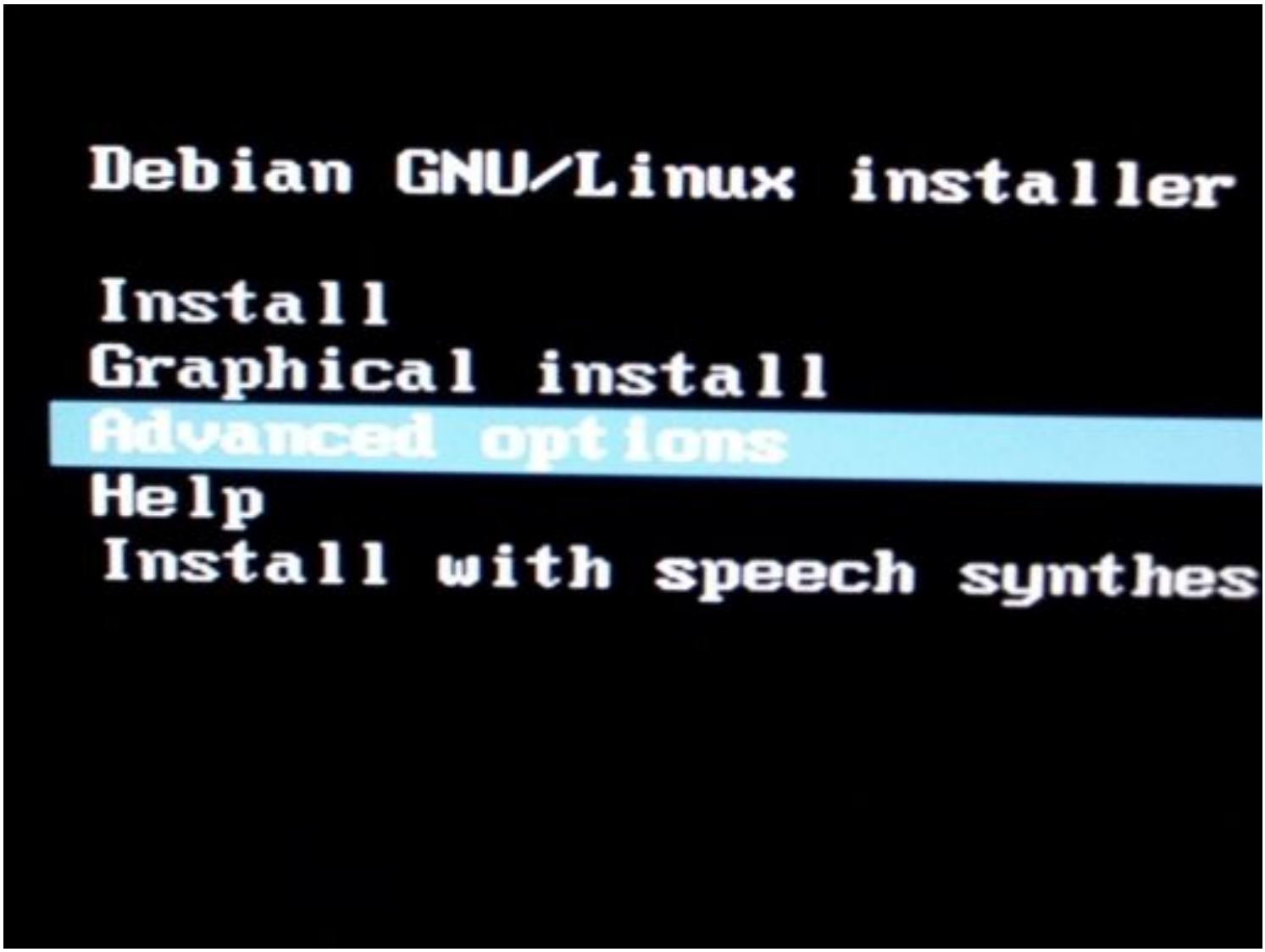
USB 3.0 ports are erratic and error prone. On Xen 4.1.2 I got it working, but only with my Wireless XBox 360 Dongle/Adapter. My Asus BT211 did not work at all, my Logitech C910 and Happauge were only able to record Audio. When I tried Xen 4.2, I got constant instabilities and BSOD's in Windows HVM. My solution was to simply not pass USB 3.0 devices.

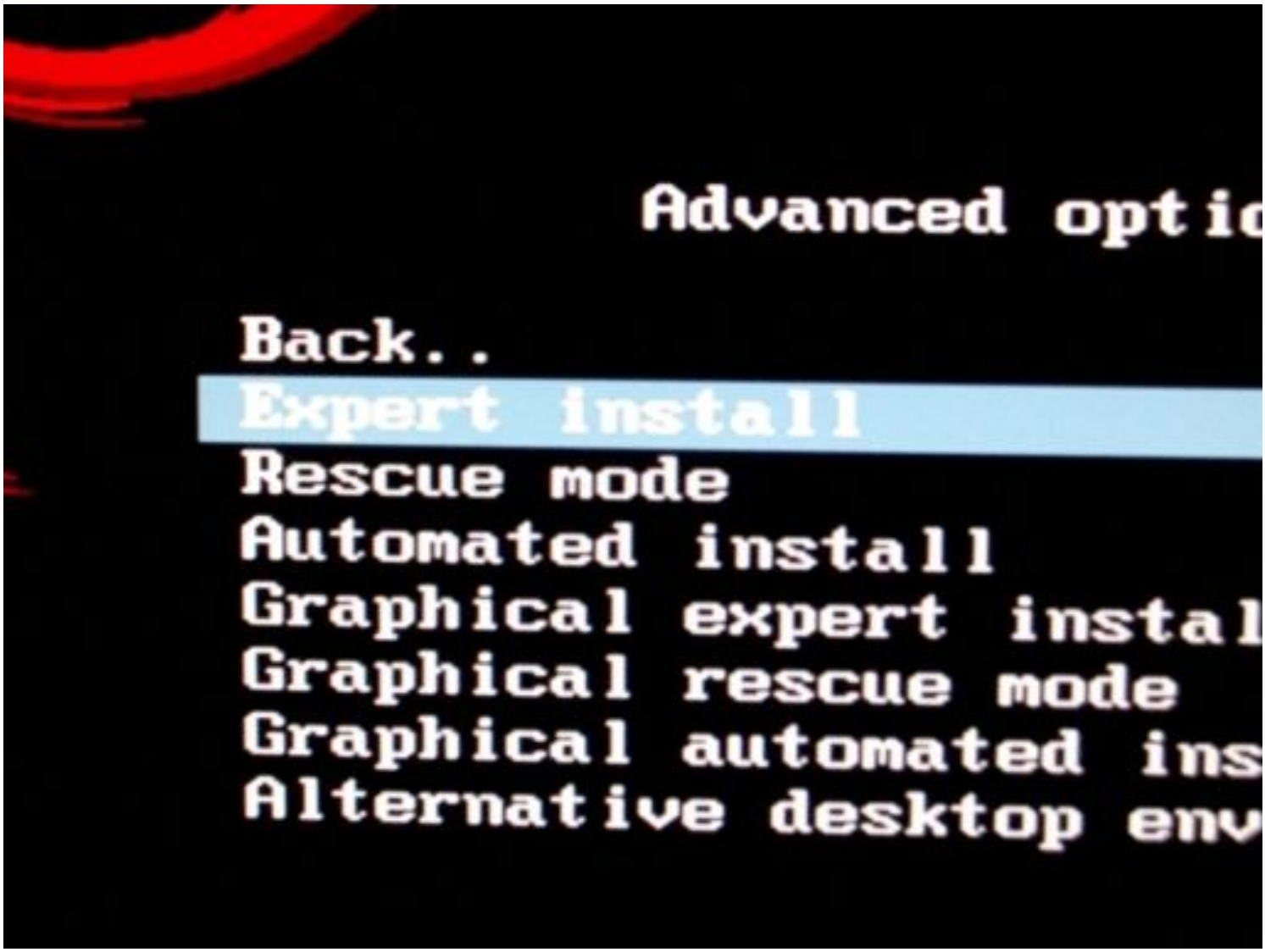
The Logitech K320 Keyboard and Nano Receiver sometimes do not work at boot time. I haven't really figured out the cause, but I recommend having SSH or a USB or PS/2 keyboard as a backup.

Installing Pure EFI Bootloaded Wheezy

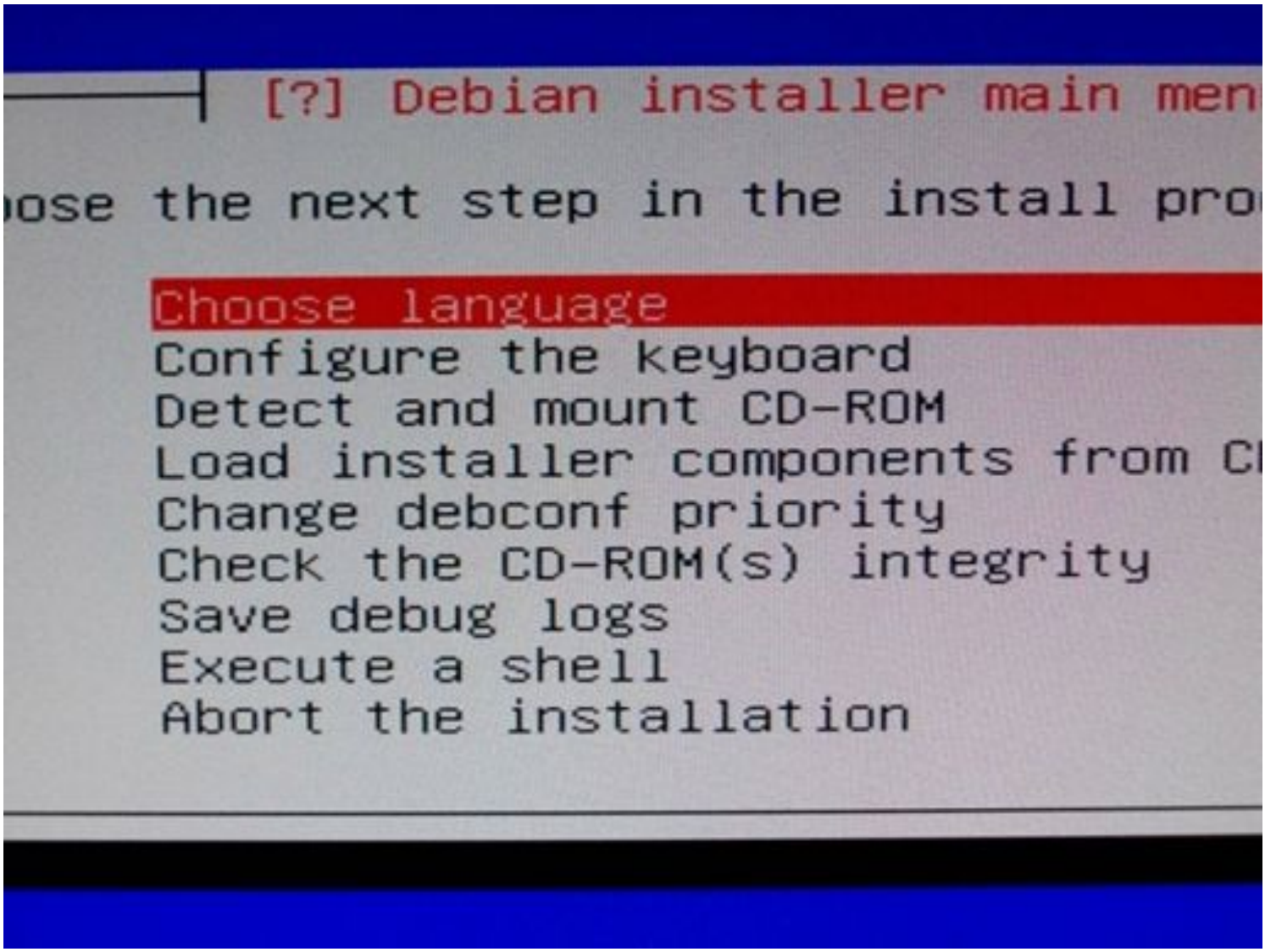
Installing Wheezy without a bootloader

One of our EFI related goals is GPT partition tables. To acheive this we have to select Expert Install from the Advanced menu:

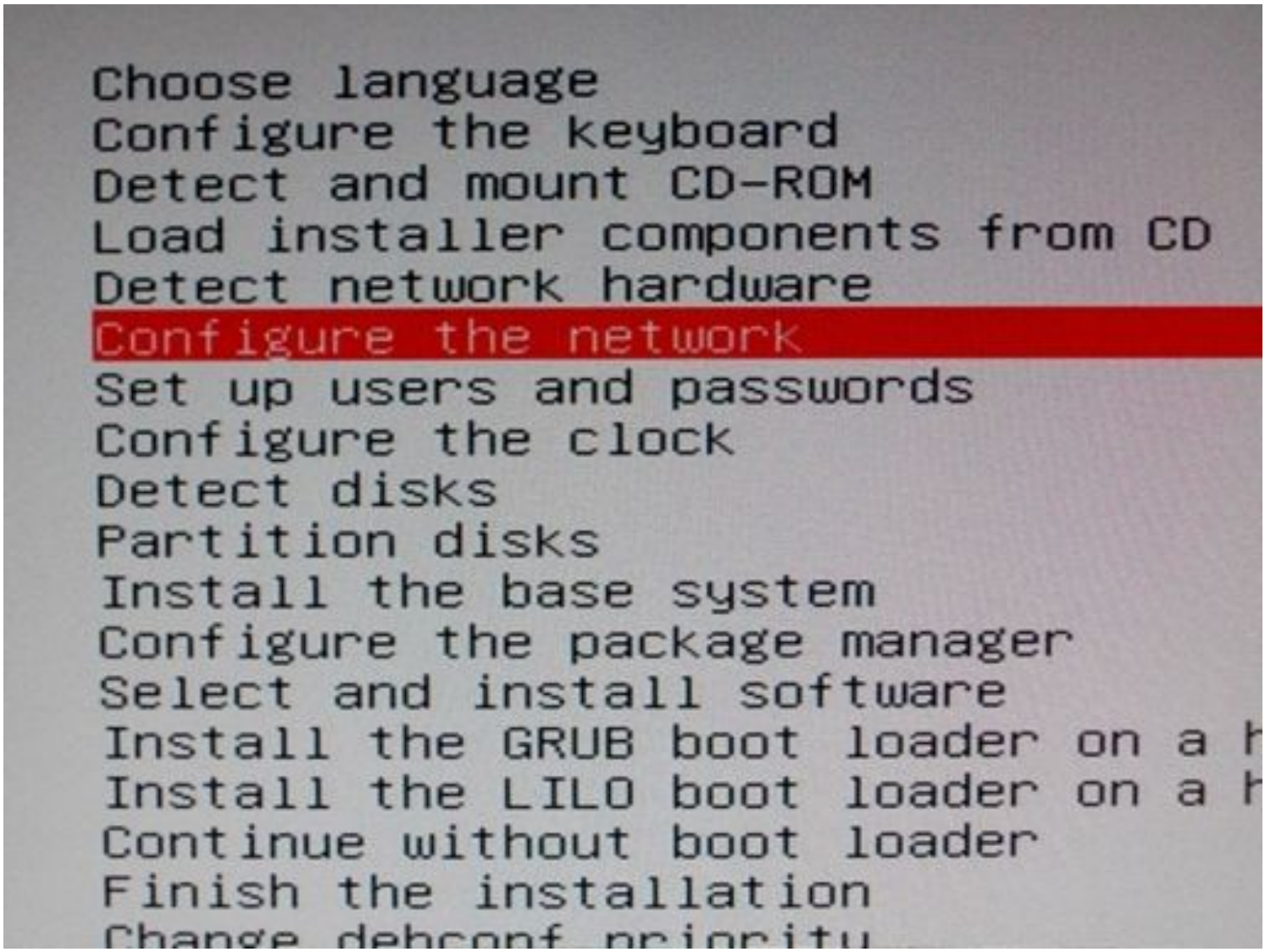

Debian GNU/Linux installer
Install
Graphical install
Advanced options
Help
Install with speech synthes



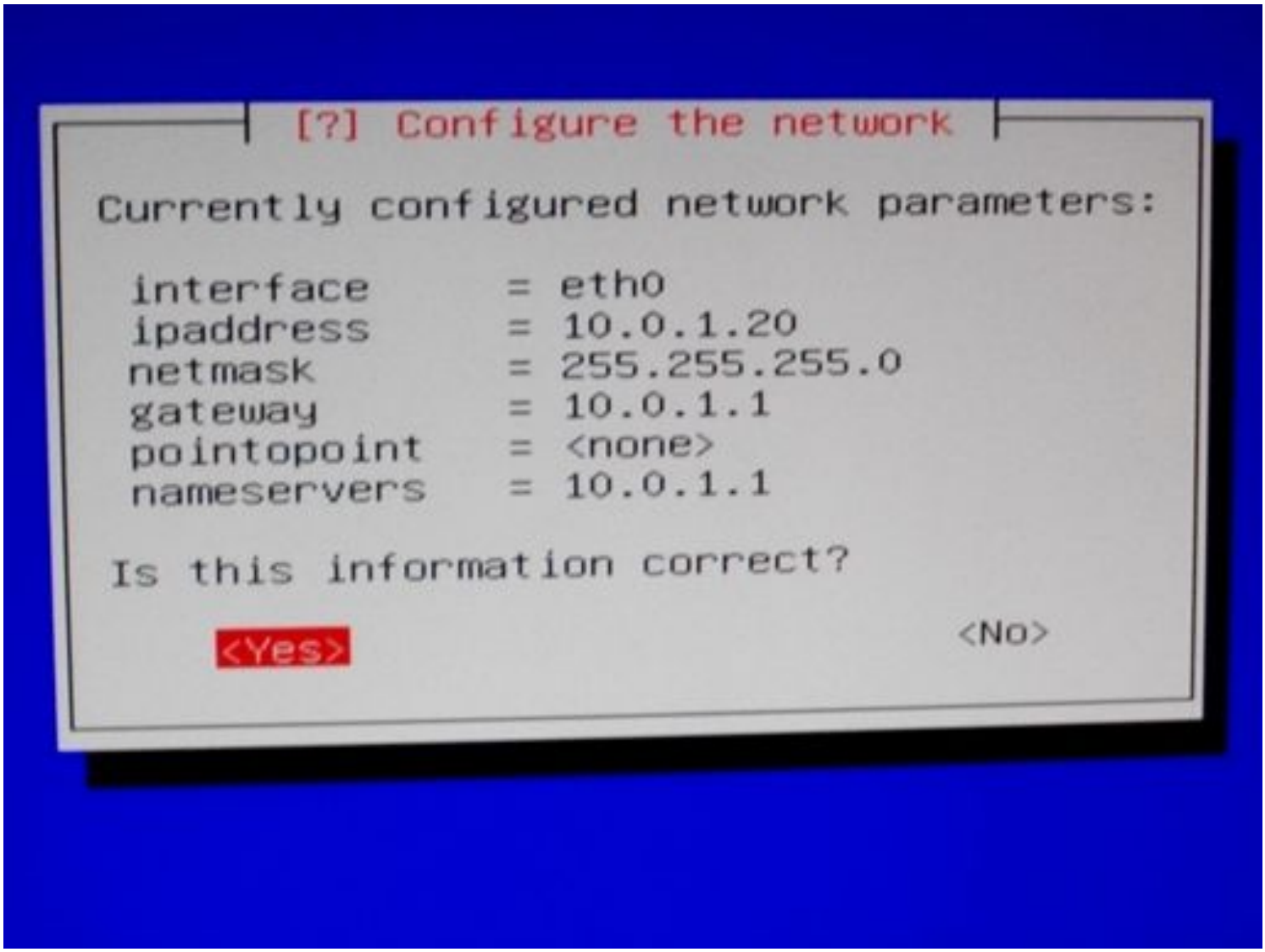
The install process is surprisingly smart, for the most part you really just confirm each screen as it pops up. We start at this screen:



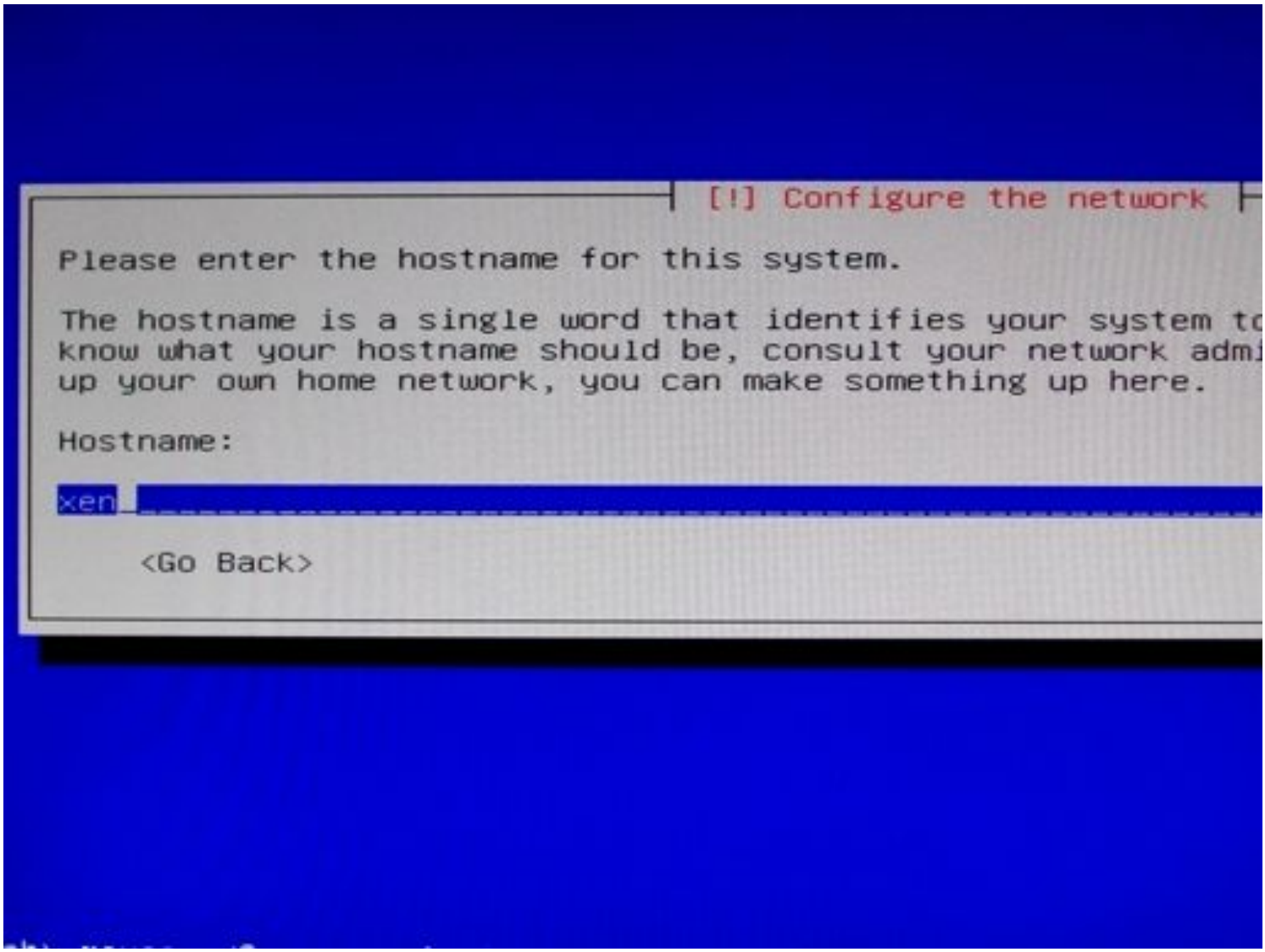
We want to confirm each dialog until we reach this menu:



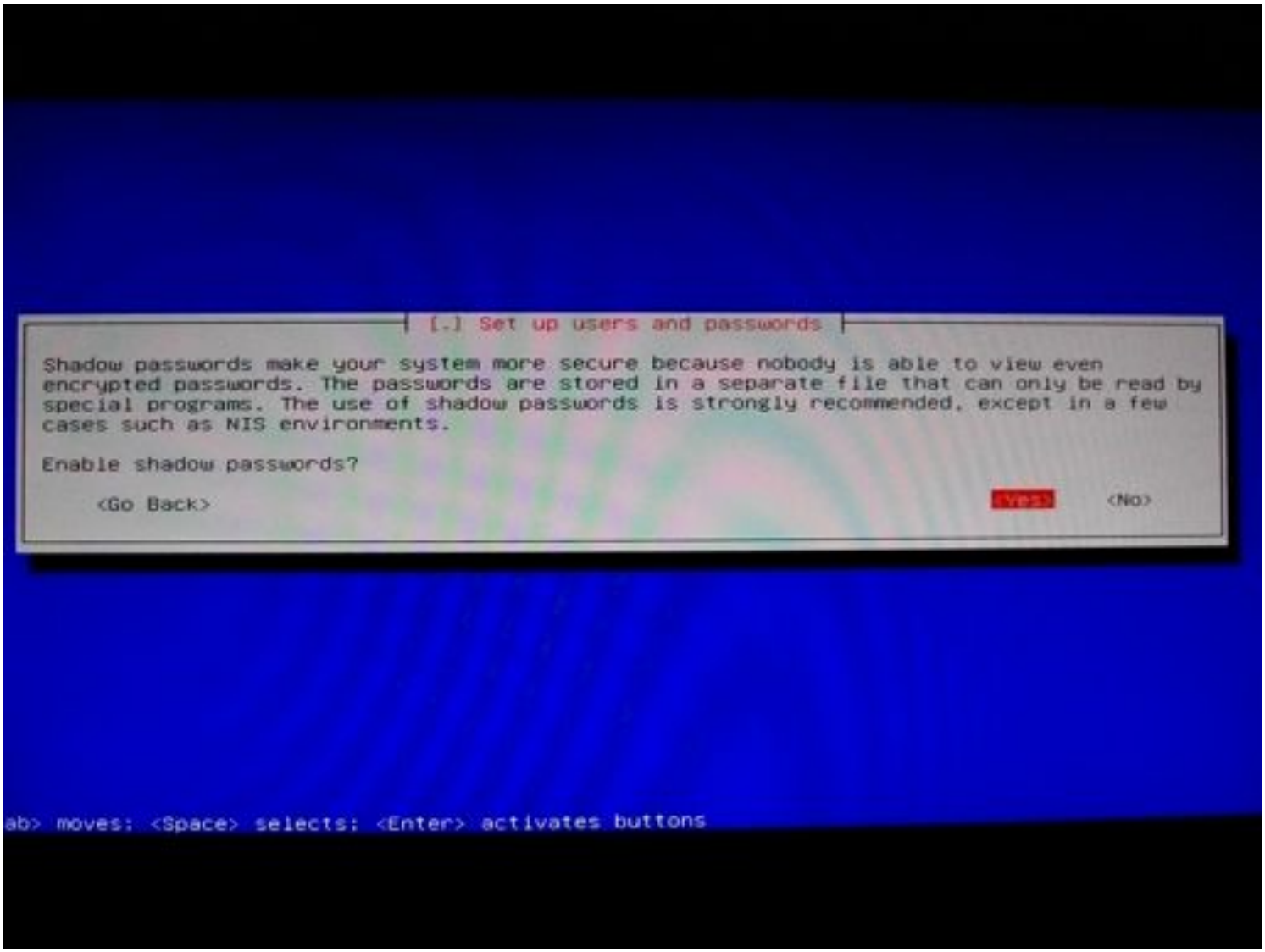
It will automatically detect the first connected ethernet port, and DHCP works but to make life easier I prefer to enter a static IP.

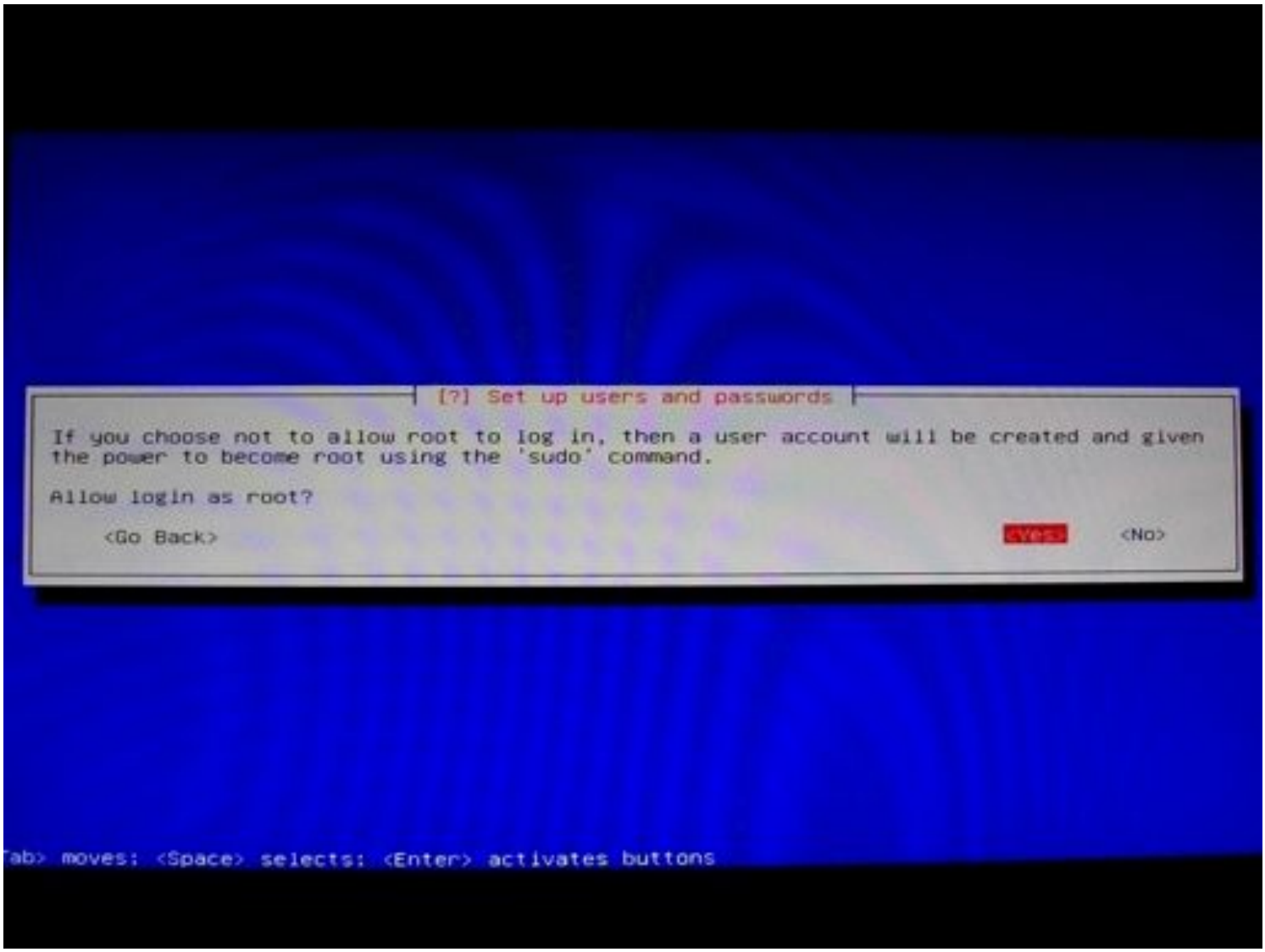


Continue confirming each screen as it passes, if you want to set the host name keep an eye out for this screen:

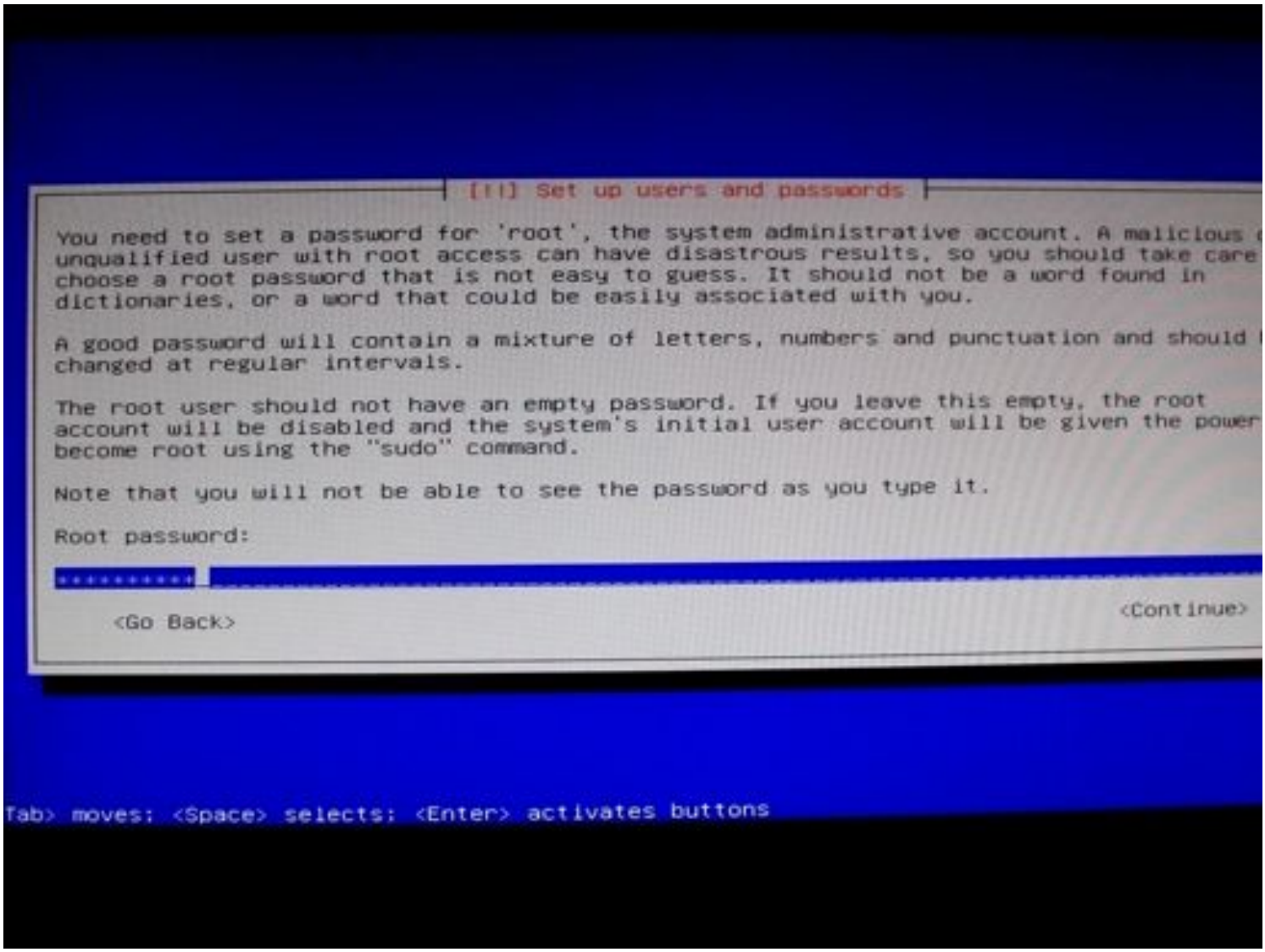


Shadow Passwords and Allow Root Login are set by default:

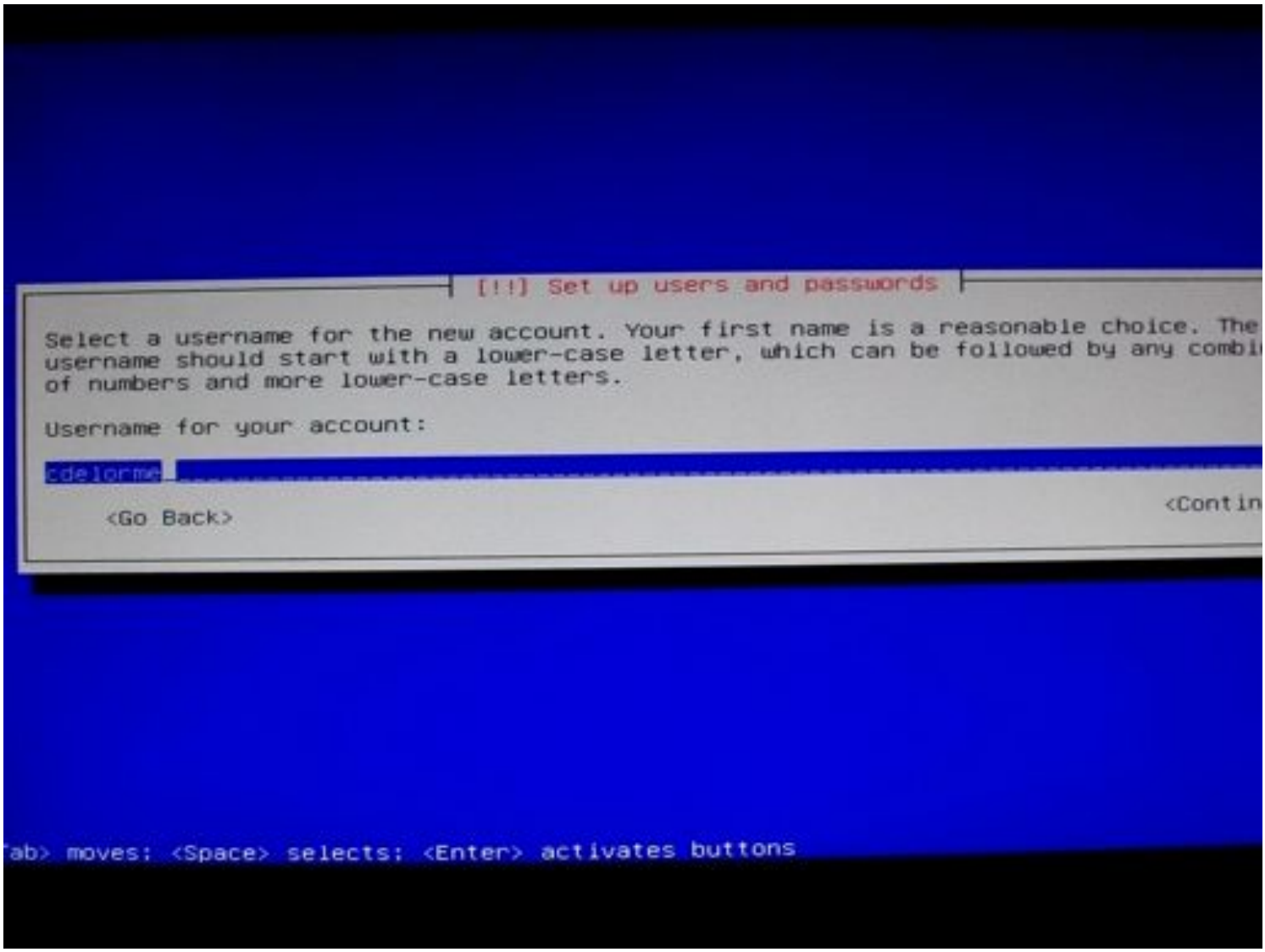

[.] Set up users and passwords
Shadow passwords make your system more secure because nobody is able to view even encrypted passwords. The passwords are stored in a separate file that can only be read by special programs. The use of shadow passwords is strongly recommended, except in a few cases such as NIS environments.
Enable shadow passwords?
<Go Back>
<Yes>
<No>
ab> moves; <Space> selects; <Enter> activates buttons



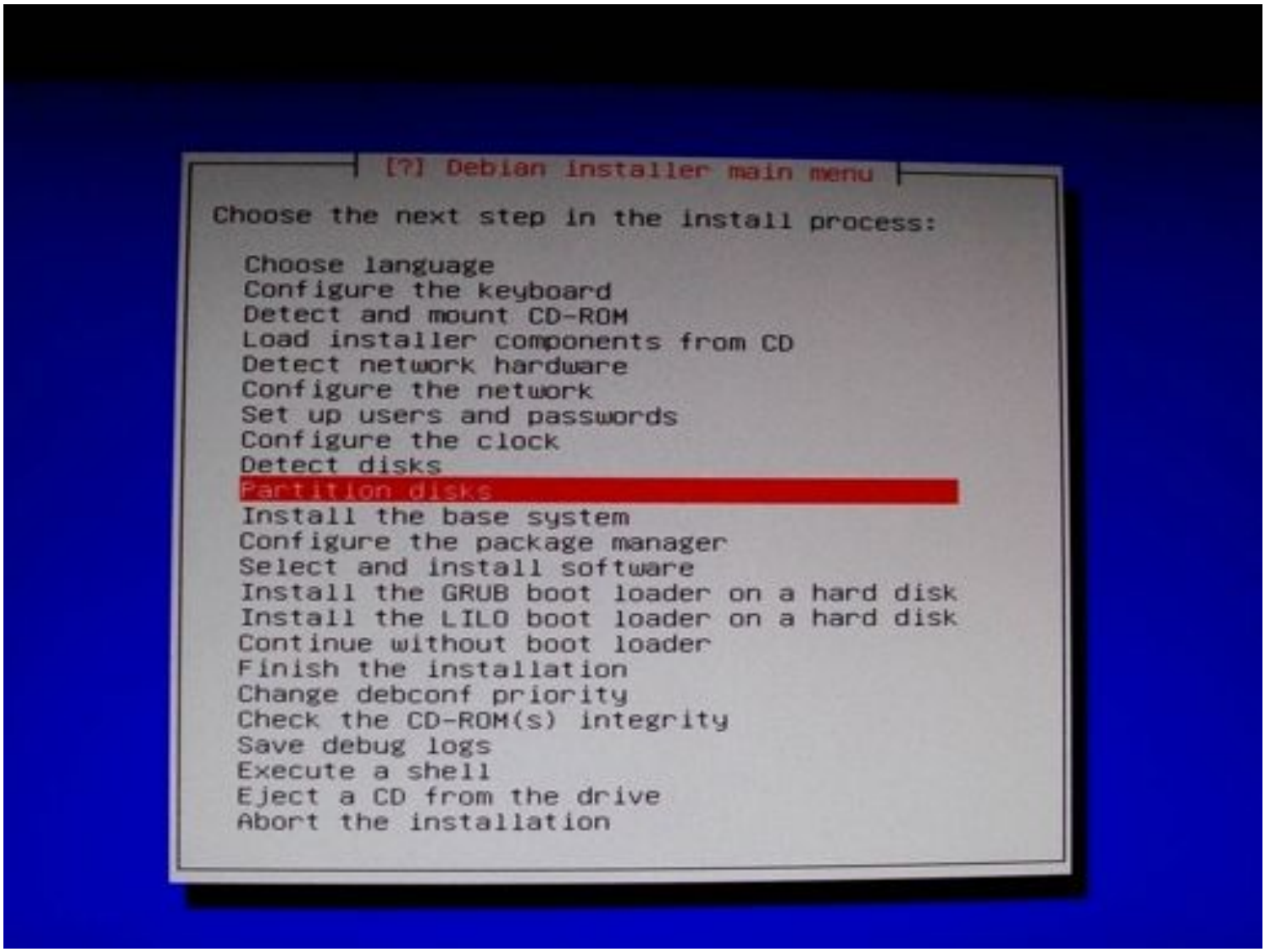
You will have to enter a password twice for both root and the new user you create (if you choose to create one):

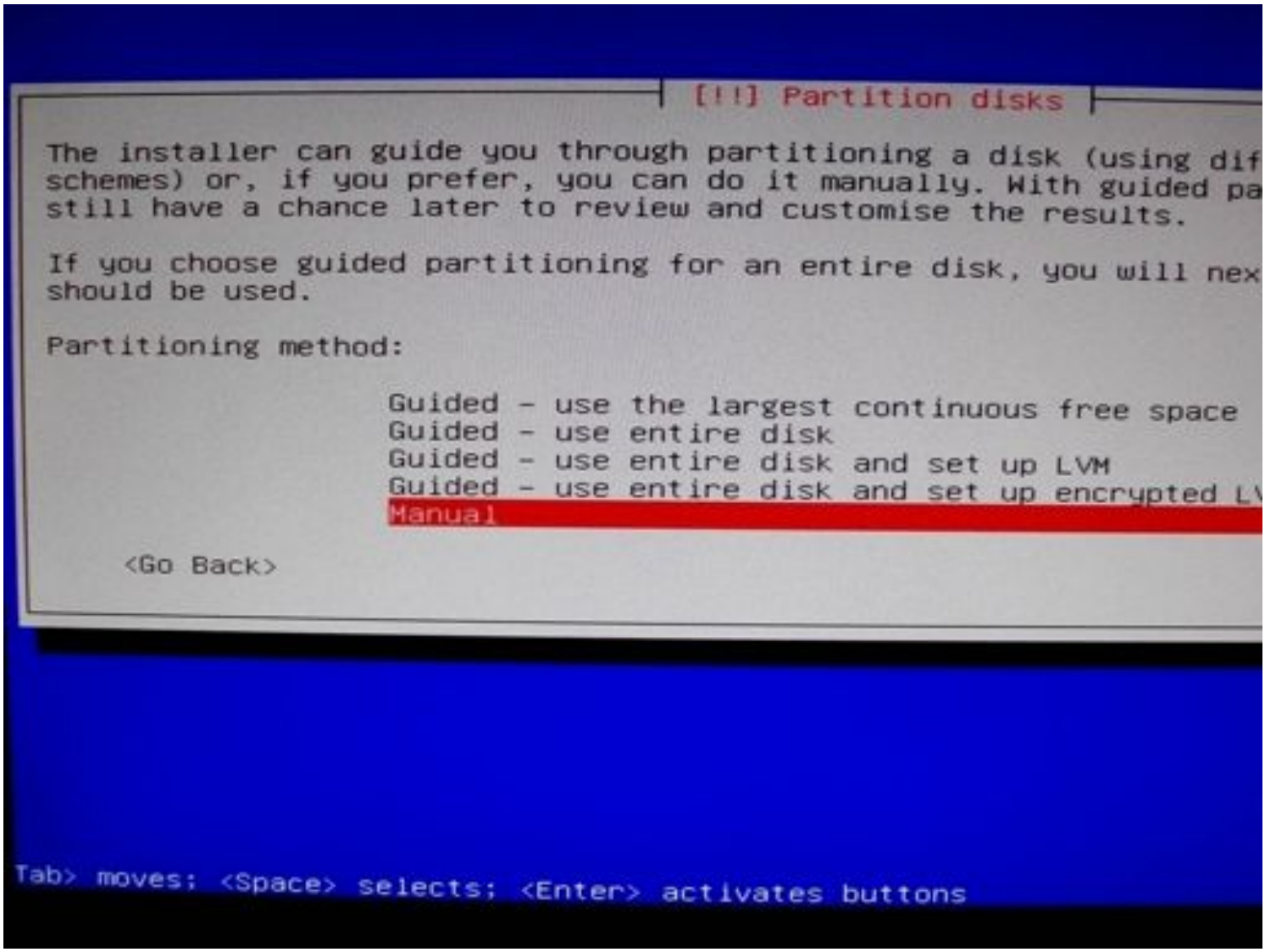

[!!] Set up users and passwords
You need to set a password for 'root', the system administrative account. A malicious
unqualified user with root access can have disastrous results, so you should take care
choose a root password that is not easy to guess. It should not be a word found in
dictionaries, or a word that could be easily associated with you.
A good password will contain a mixture of letters, numbers and punctuation and should
changed at regular intervals.
The root user should not have an empty password. If you leave this empty, the root
account will be disabled and the system's initial user account will be given the power
become root using the "sudo" command.
Note that you will not be able to see the password as you type it.
Root password:
<Go Back>
<Continue>
Tab> moves; <Space> selects; <Enter> activates buttons

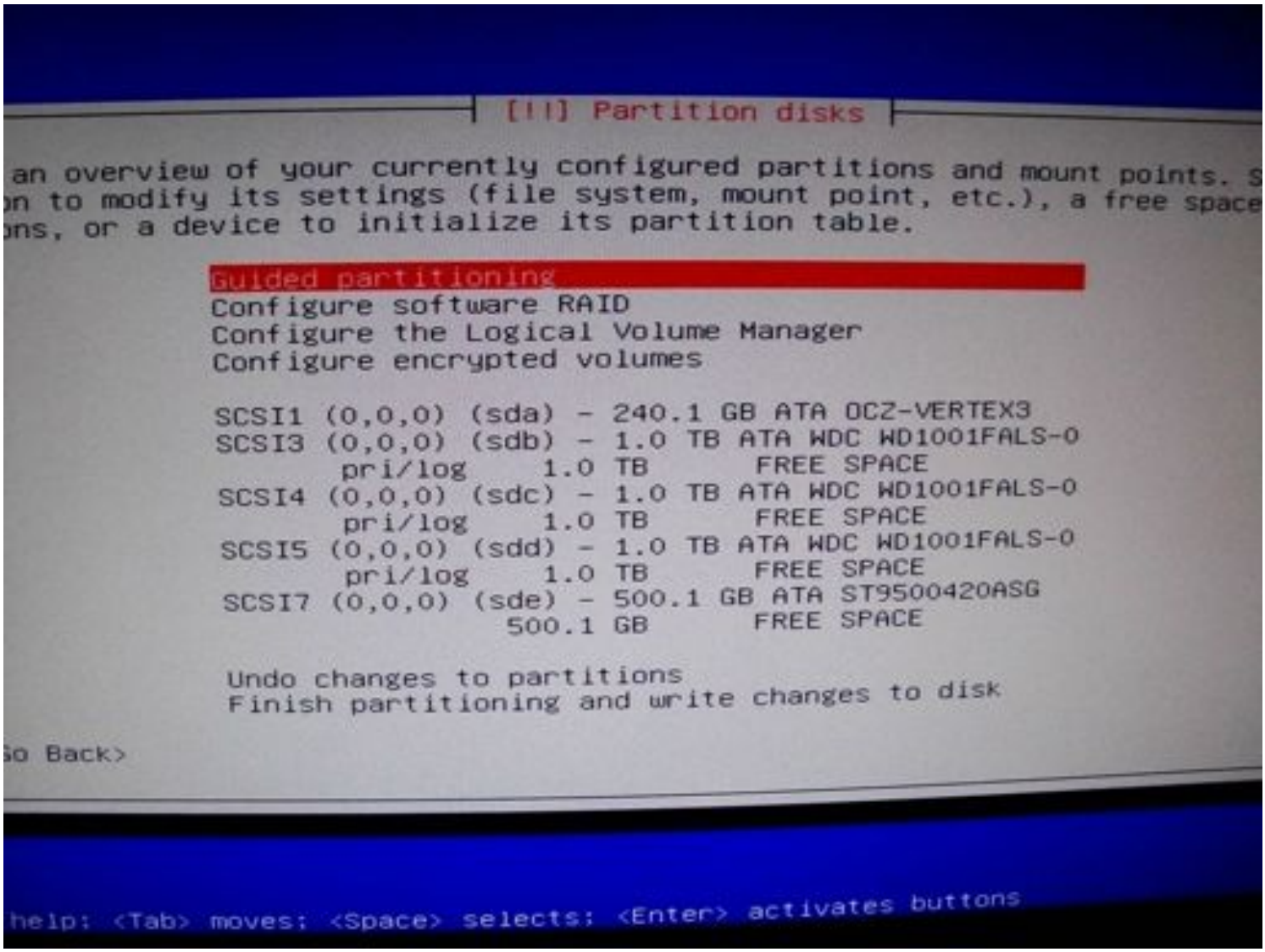

[!!] Set up users and passwords
A user account will be created for you to use instead of the root account for non-administrative activities.
Please enter the real name of this user. This information will be used for instance default origin for emails sent by this user as well as any program which displays t the user's real name. Your full name is a reasonable choice.
Full name for the new user:
Casey DeLorme
<Go Back>
<Continu
ab> moves; <Space> selects; <Enter> activates buttons



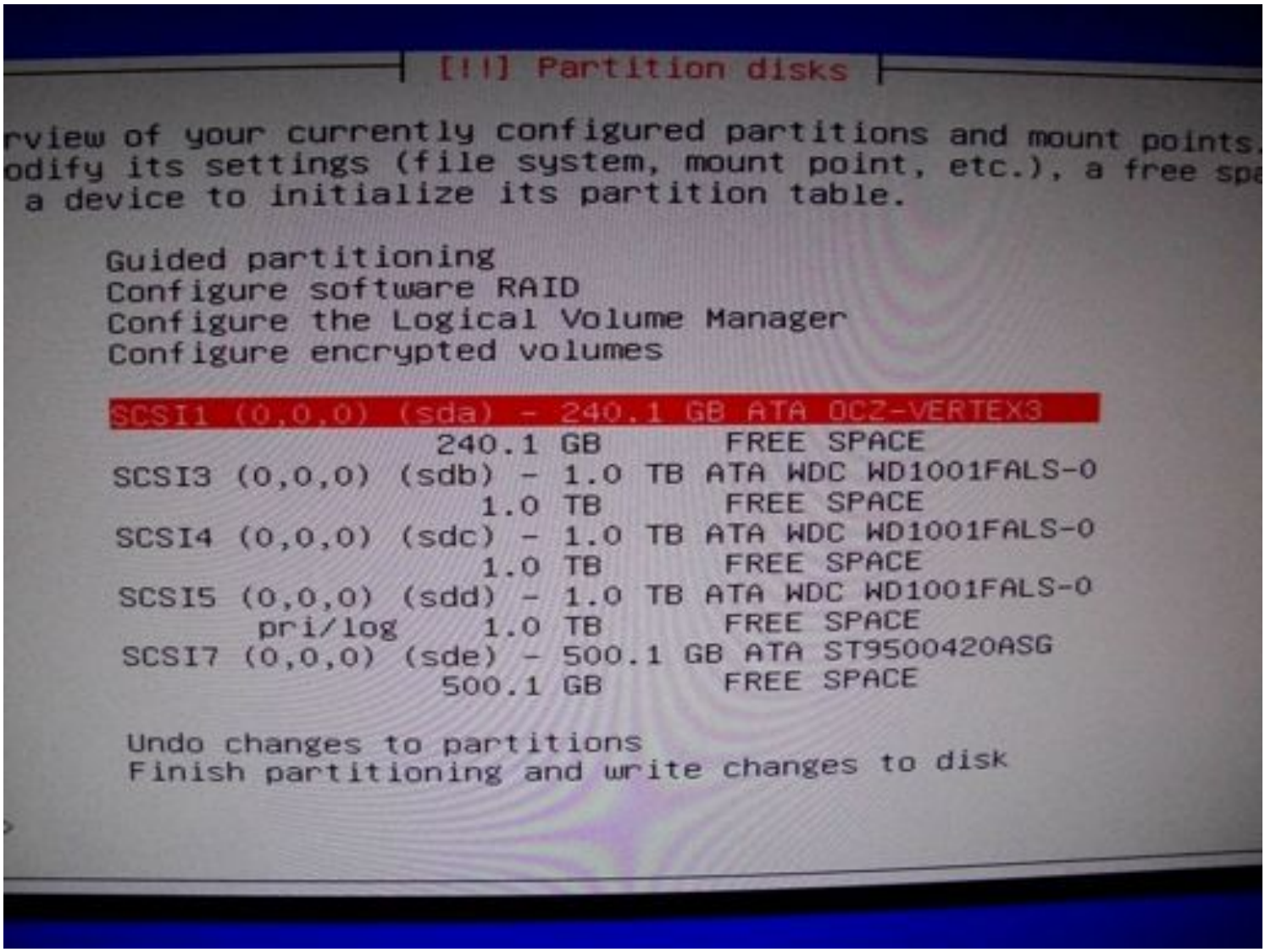
Continue confirming each screen until you reach disk partitioning, at this stage select manual:

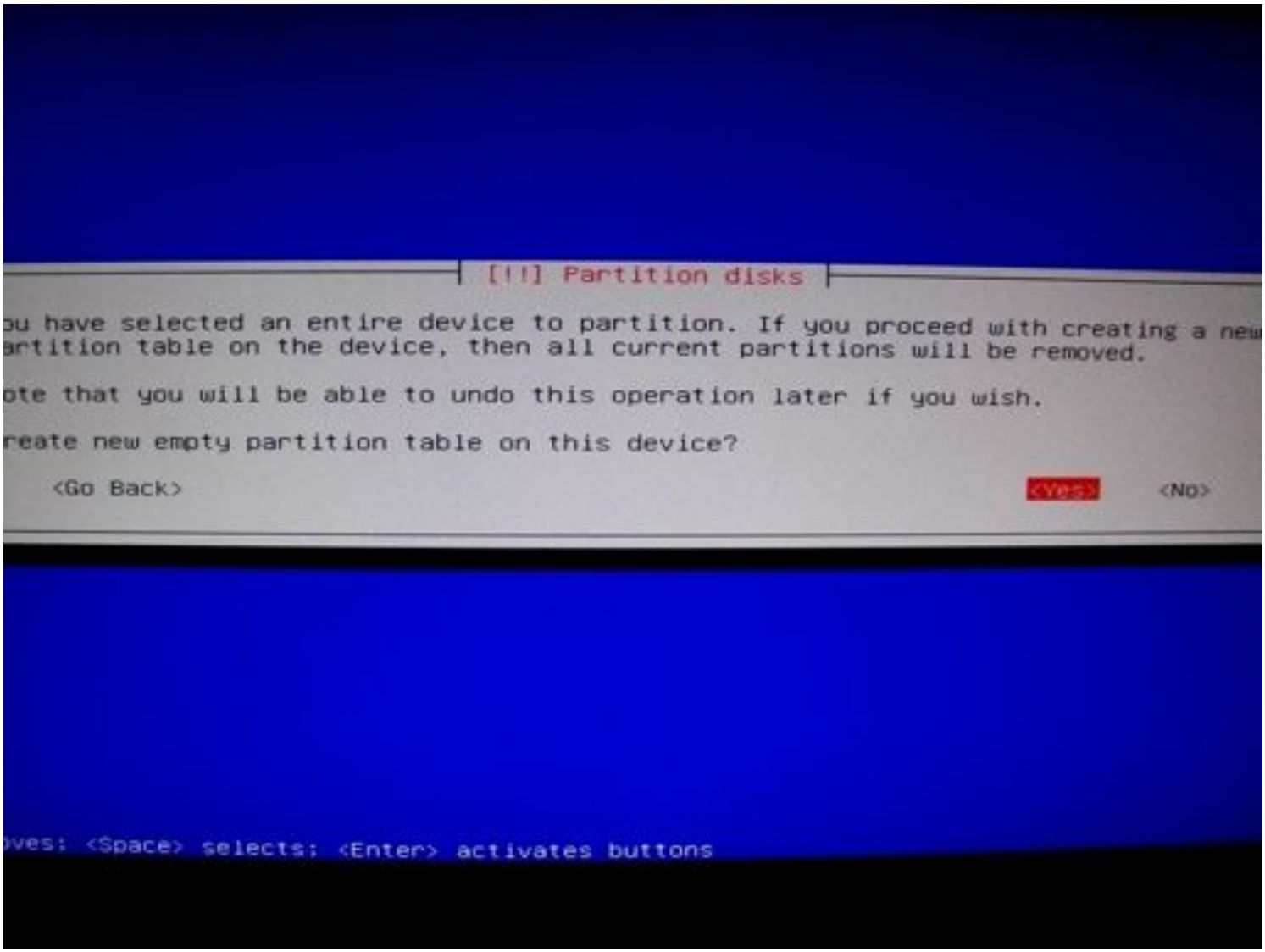

[?] Debian installer main menu
Choose the next step in the install process:
Choose language
Configure the keyboard
Detect and mount CD-ROM
Load installer components from CD
Detect network hardware
Configure the network
Set up users and passwords
Configure the clock
Detect disks
Partition disks
Install the base system
Configure the package manager
Select and install software
Install the GRUB boot loader on a hard disk
Install the LILO boot loader on a hard disk
Continue without boot loader
Finish the installation
Change debconf priority
Check the CD-ROM(s) integrity
Save debug logs
Execute a shell
Eject a CD from the drive
Abort the installation

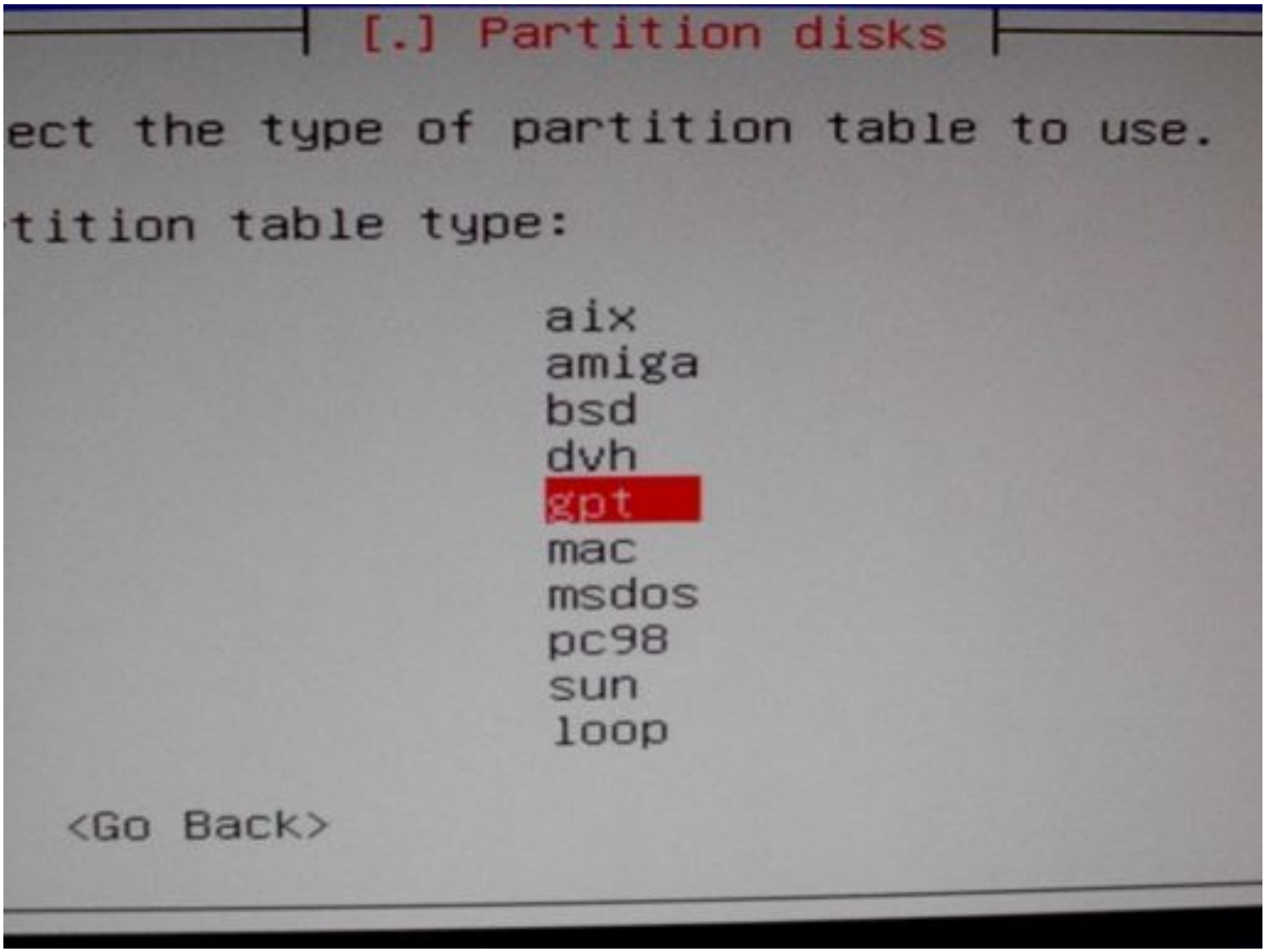

[!!] Partition disks
The installer can guide you through partitioning a disk (using dif
schemes) or, if you prefer, you can do it manually. With guided pa
still have a chance later to review and customise the results.
If you choose guided partitioning for an entire disk, you will nex
should be used.
Partitioning method:
Guided - use the largest continuous free space
Guided - use entire disk
Guided - use entire disk and set up LVM
Guided - use entire disk and set up encrypted LV
Manual
<Go Back>
Tab> moves; <Space> selects; <Enter> activates buttons



Select each disk you intend to use, and specify the partition table as GPT, the default will be msdos:


[!!] Partition disks
rview of your currently configured partitions and mount points.
odify its settings (file system, mount point, etc.), a free spa
a device to initialize its partition table.
Guided partitioning
Configure software RAID
Configure the Logical Volume Manager
Configure encrypted volumes
SCSI1 (0,0,0) (sda) - 240.1 GB ATA OCZ-VERTEX3
240.1 GB FREE SPACE
SCSI3 (0,0,0) (sdb) - 1.0 TB ATA WDC WD1001FALS-0
1.0 TB FREE SPACE
SCSI4 (0,0,0) (sdc) - 1.0 TB ATA WDC WD1001FALS-0
1.0 TB FREE SPACE
SCSI5 (0,0,0) (sdd) - 1.0 TB ATA WDC WD1001FALS-0
pri/log 1.0 TB FREE SPACE
SCSI7 (0,0,0) (sde) - 500.1 GB ATA ST9500420ASG
500.1 GB FREE SPACE
Undo changes to partitions
Finish partitioning and write changes to disk

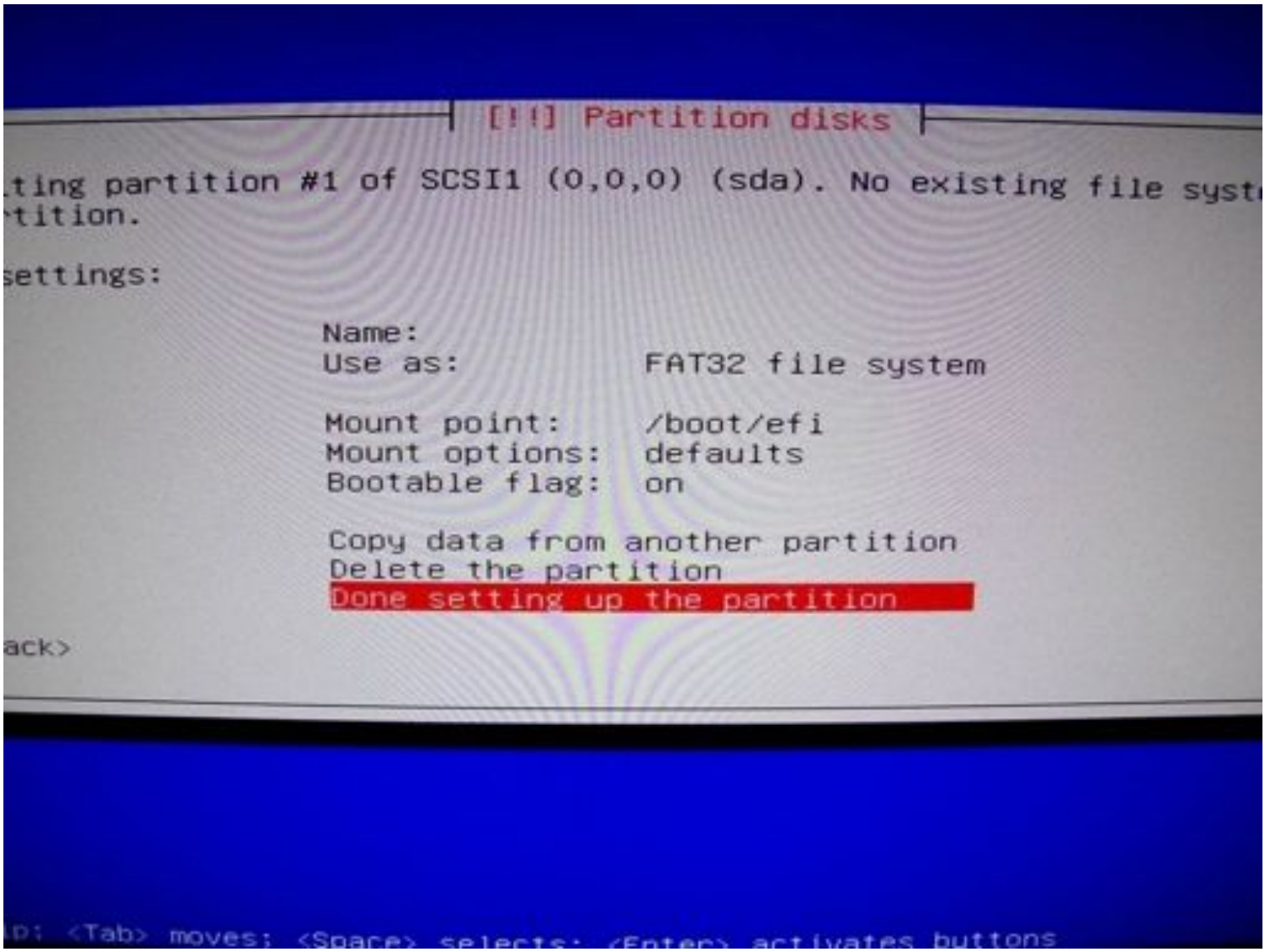

[!!] Partition disks
ou have selected an entire device to partition. If you proceed with creating a new
artition table on the device, then all current partitions will be removed.
ote that you will be able to undo this operation later if you wish.
reate new empty partition table on this device?
<Go Back>
<Yes>
<No>
ves; <Space> selects; <Enter> activates buttons



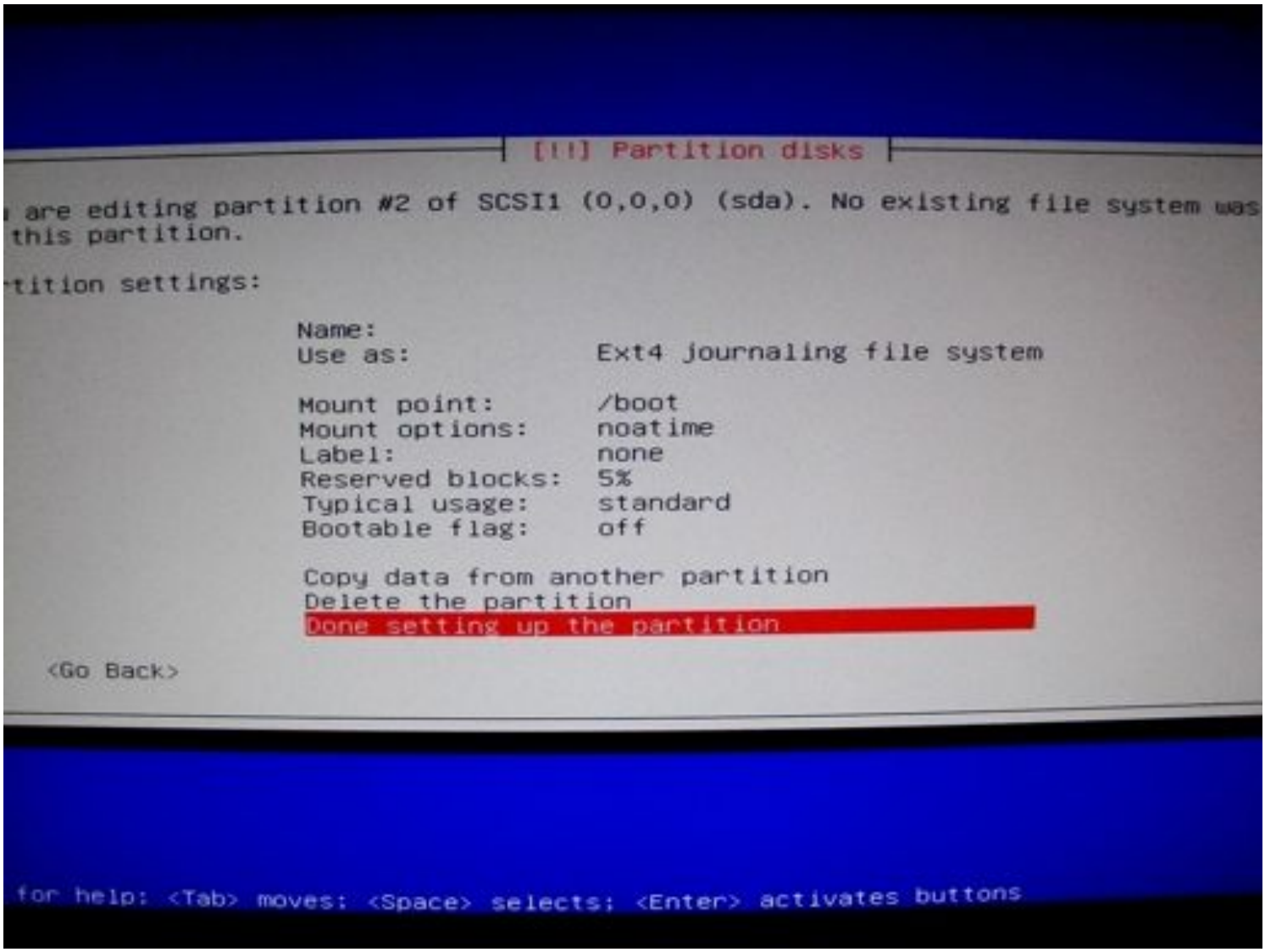
My configuration is a big confusing as I have a lot of drives connected. I am using a 240GB SSD as my main drive, a 500 GB Laptop hard drive to store backups, and 3x 1TB Western Digital Cavier Black drives for storage.

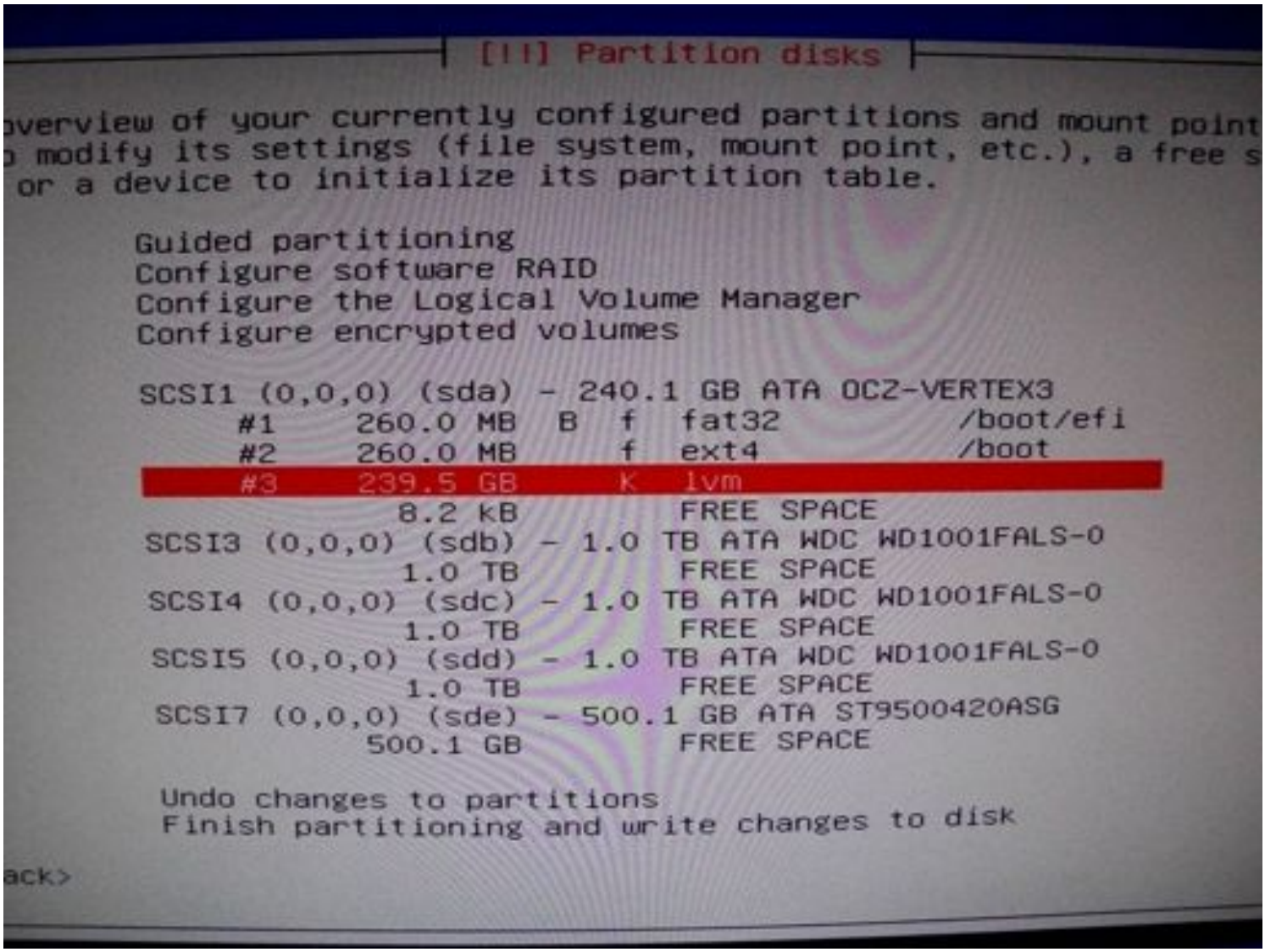
My SSD has three partitions, 260MB Fat32 for EFI, 260MB ext4 for /boot, and the remainder to LVM. My 500 GB Laptop drive contains a single LVM partition, and the three 1TB Drives are setup as a RAID 5 2TB Storage area, with a single LVM Partition.

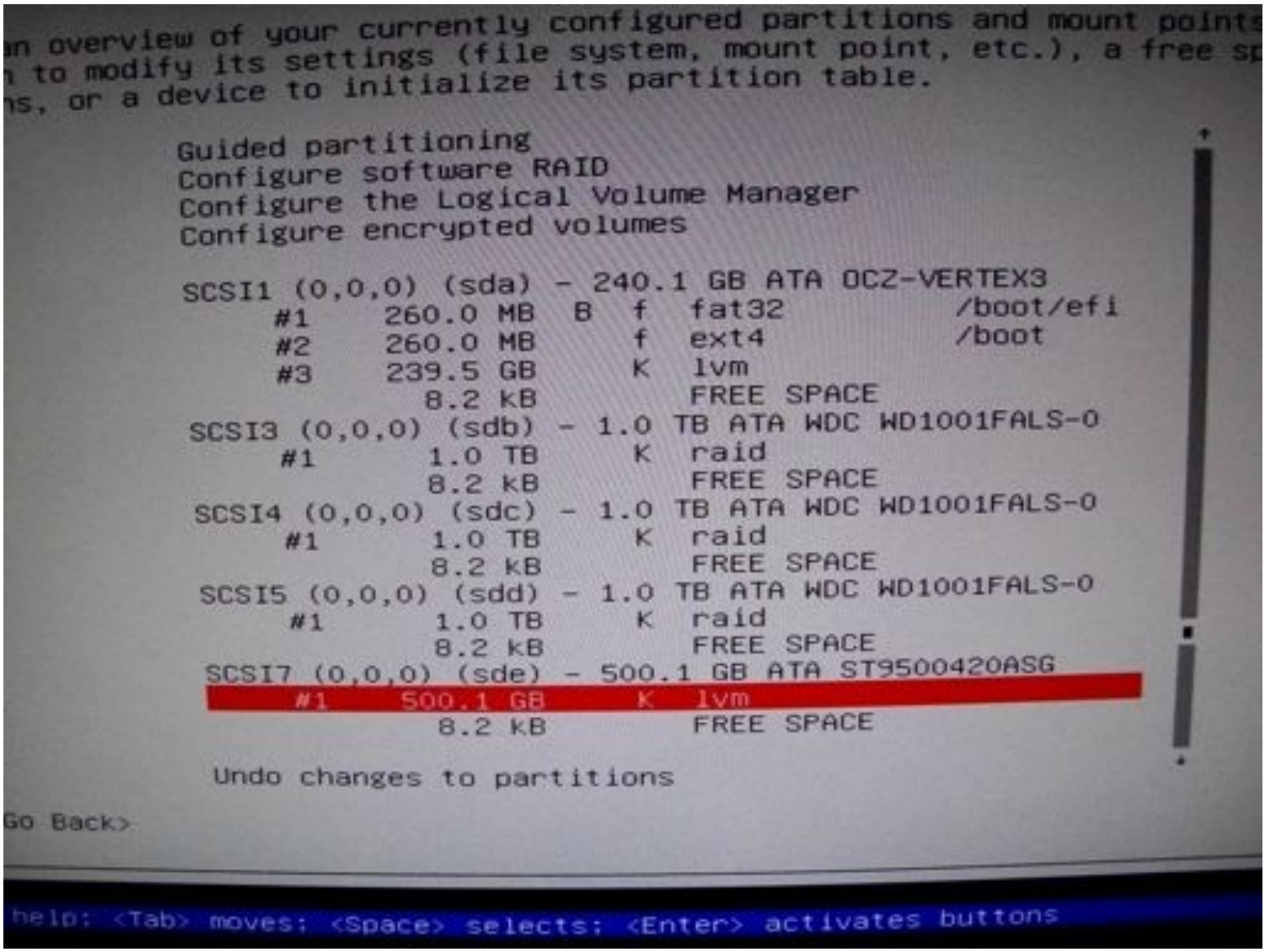
For EFI you need a FAT32 partition, the smallest size is around 260 Megabytes, unless you plan on doing a lot more with it you shouldn't need more than that. Be sure you mark this partition as bootable, and for Debian you want to set its mount point manually to /boot/efi:



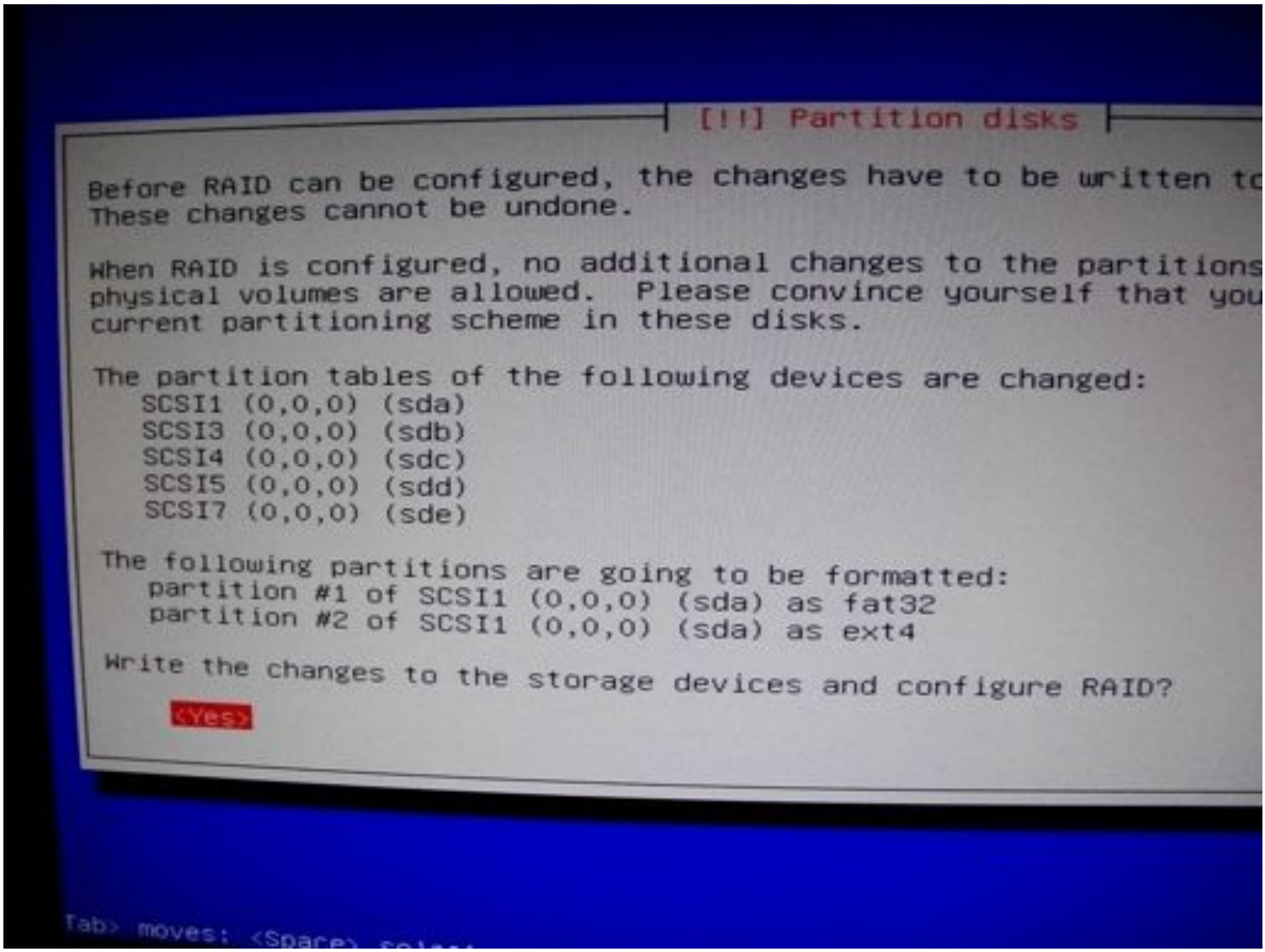
We also need a partition for your boot directory. This should be a linux format, ext3 or ext4, for SSD owners I recommend ext4. The mount point should be /boot, and this is the partition that allows us to boot our OS off of the LVM partitions.

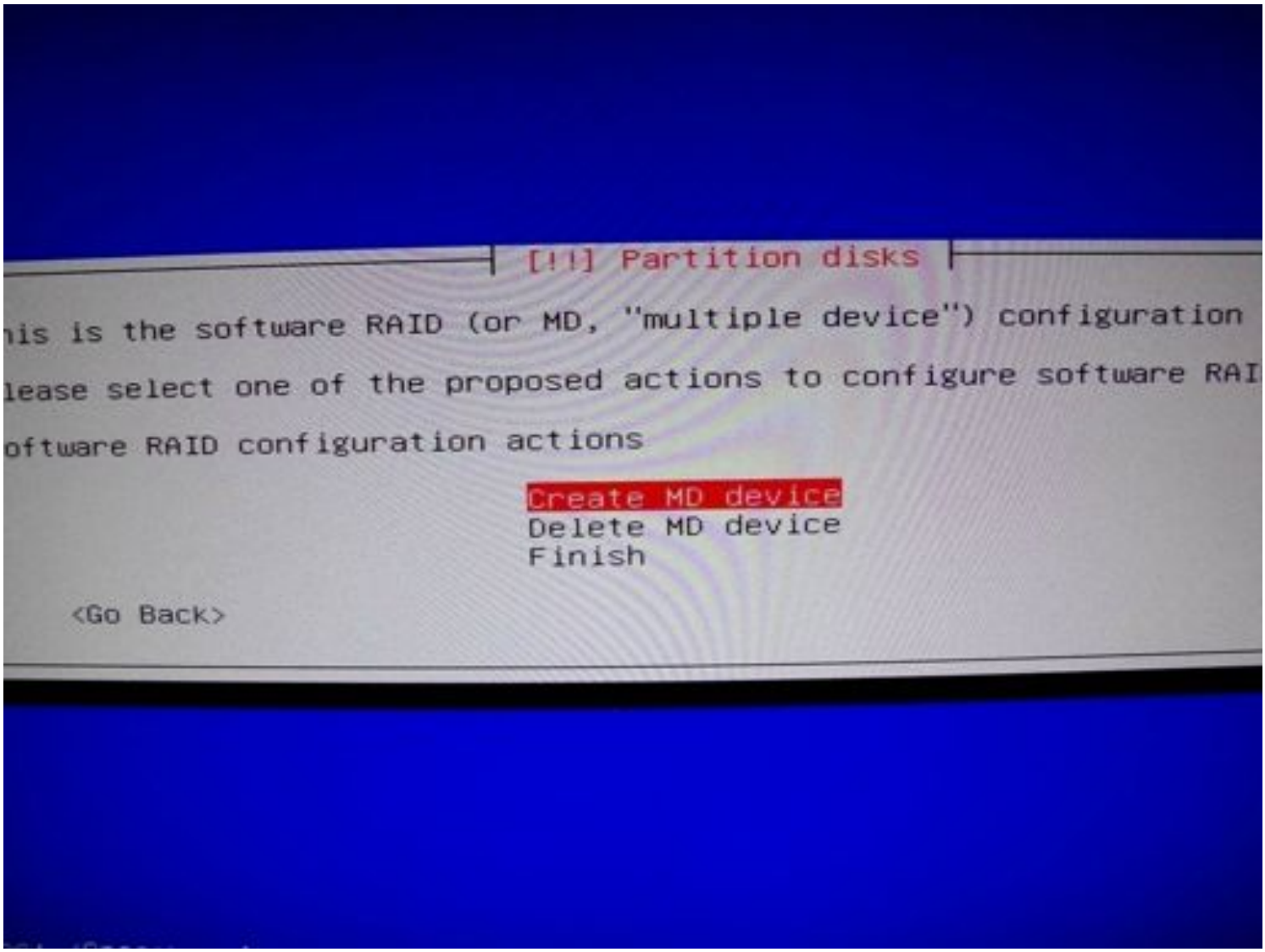

[!!] Partition disks
are editing partition #2 of SCSI1 (0,0,0) (sda). No existing file system was
this partition.
tition settings:
Name:
Use as: Ext4 journaling file system
Mount point: /boot
Mount options: noatime
Label: none
Reserved blocks: 5%
Typical usage: standard
Bootable flag: off
Copy data from another partition
Delete the partition
Done setting up the partition
<Go Back>
for help: <Tab> moves; <Space> selects; <Enter> activates buttons


[!!] Partition disks
overview of your currently configured partitions and mount point
o modify its settings (file system, mount point, etc.), a free s
or a device to initialize its partition table.
Guided partitioning
Configure software RAID
Configure the Logical Volume Manager
Configure encrypted volumes
SCSI1 (0,0,0) (sda) - 240.1 GB ATA OCZ-VERTEX3
#1 260.0 MB B f fat32 /boot/efi
#2 260.0 MB f ext4 /boot
#3 239.5 GB K lvm
8.2 kB FREE SPACE
SCSI3 (0,0,0) (sdb) - 1.0 TB ATA WDC WD1001FALS-0
1.0 TB FREE SPACE
SCSI4 (0,0,0) (sdc) - 1.0 TB ATA WDC WD1001FALS-0
1.0 TB FREE SPACE
SCSI5 (0,0,0) (sdd) - 1.0 TB ATA WDC WD1001FALS-0
1.0 TB FREE SPACE
SCSI7 (0,0,0) (sde) - 500.1 GB ATA ST9500420ASG
500.1 GB FREE SPACE
Undo changes to partitions
Finish partitioning and write changes to disk
ack>

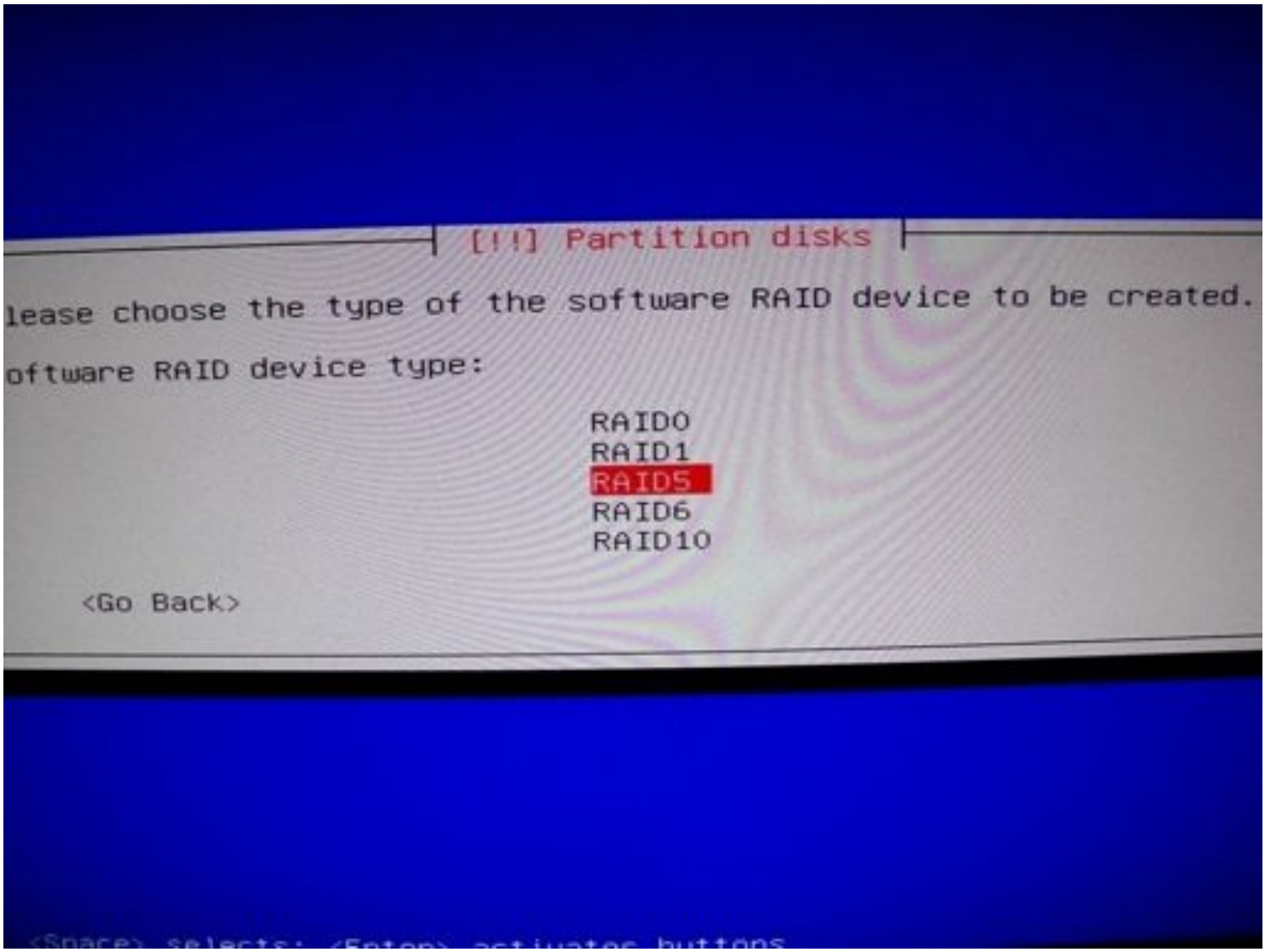


For setting up the raid, we have to confirm configuration changes so far, and select create md:

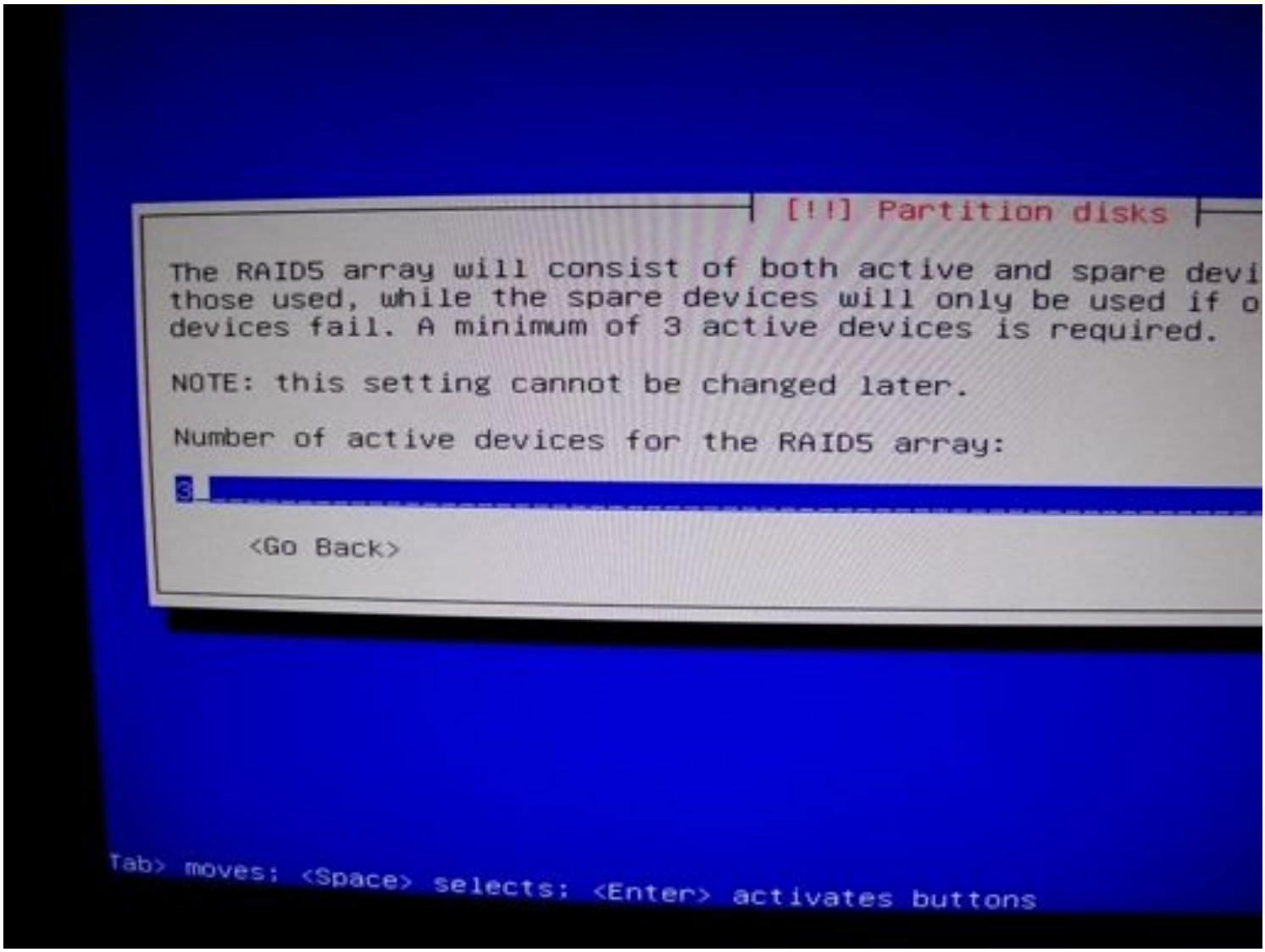

[!!] Partition disks
Before RAID can be configured, the changes have to be written to
These changes cannot be undone.
When RAID is configured, no additional changes to the partitions
physical volumes are allowed. Please convince yourself that you
current partitioning scheme in these disks.
The partition tables of the following devices are changed:
SCSI1 (0,0,0) (sda)
SCSI3 (0,0,0) (sdb)
SCSI4 (0,0,0) (sdc)
SCSI5 (0,0,0) (sdd)
SCSI7 (0,0,0) (sde)
The following partitions are going to be formatted:
partition #1 of SCSI1 (0,0,0) (sda) as fat32
partition #2 of SCSI1 (0,0,0) (sda) as ext4
Write the changes to the storage devices and configure RAID?
<Yes>
Tab> moves; <Space>

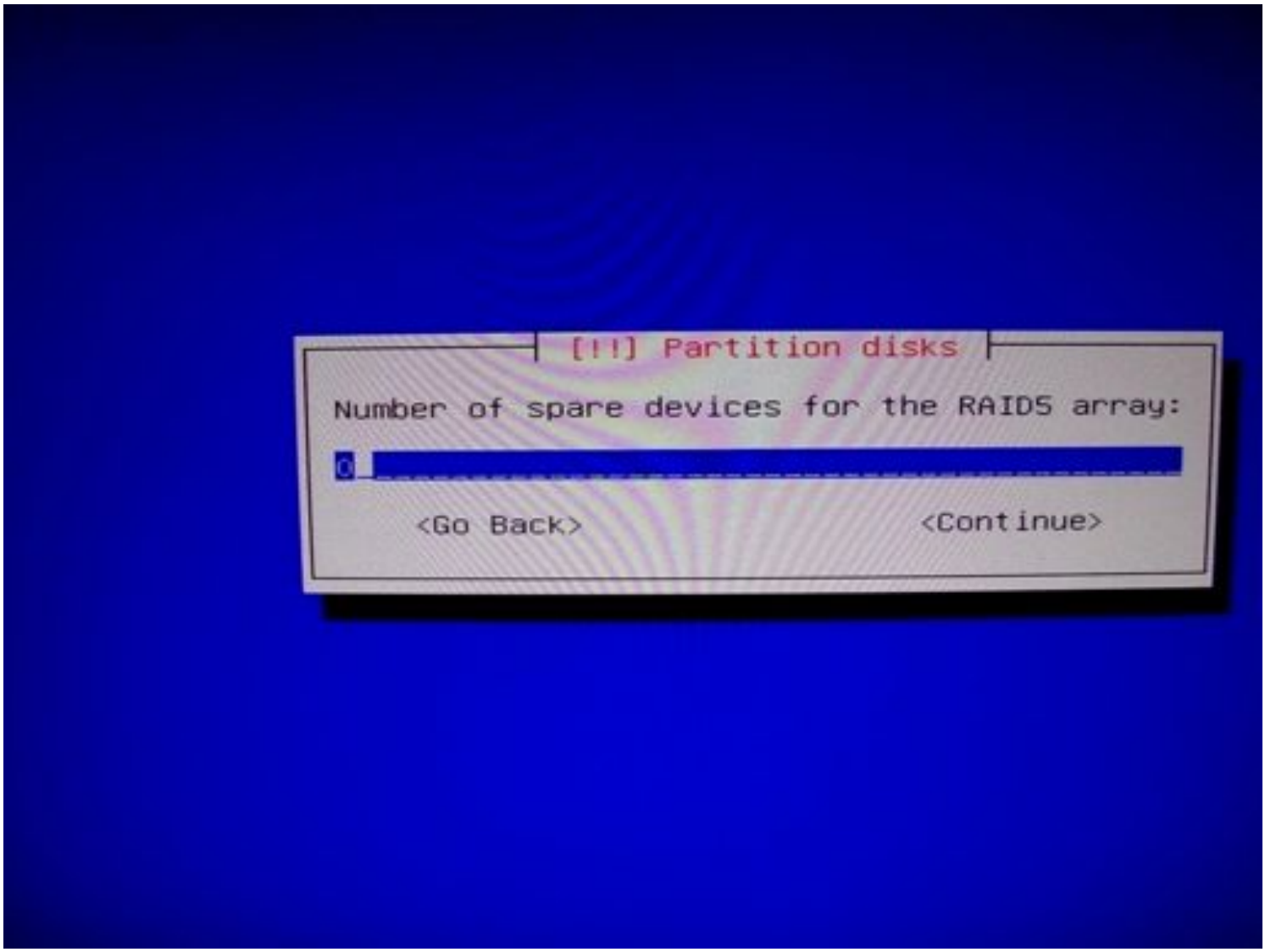


We can select a RAID type, there are many kinds, if you aren't familiar check out the RAID on Wikipedia:

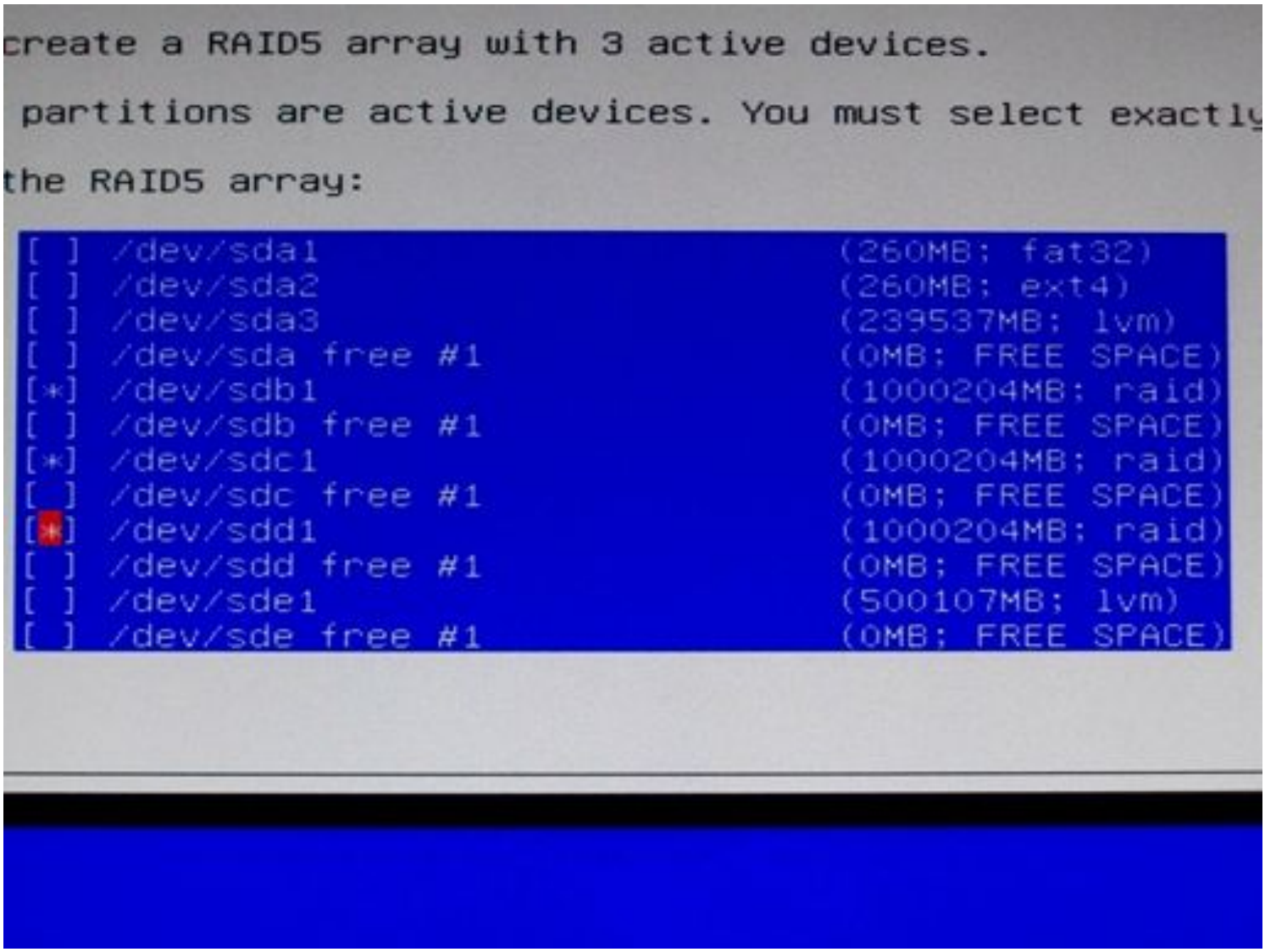


The configuration will ask us how many disks we are using in our configuration, and if there are any spare disks (disks you have not in use but there as immediate backups):

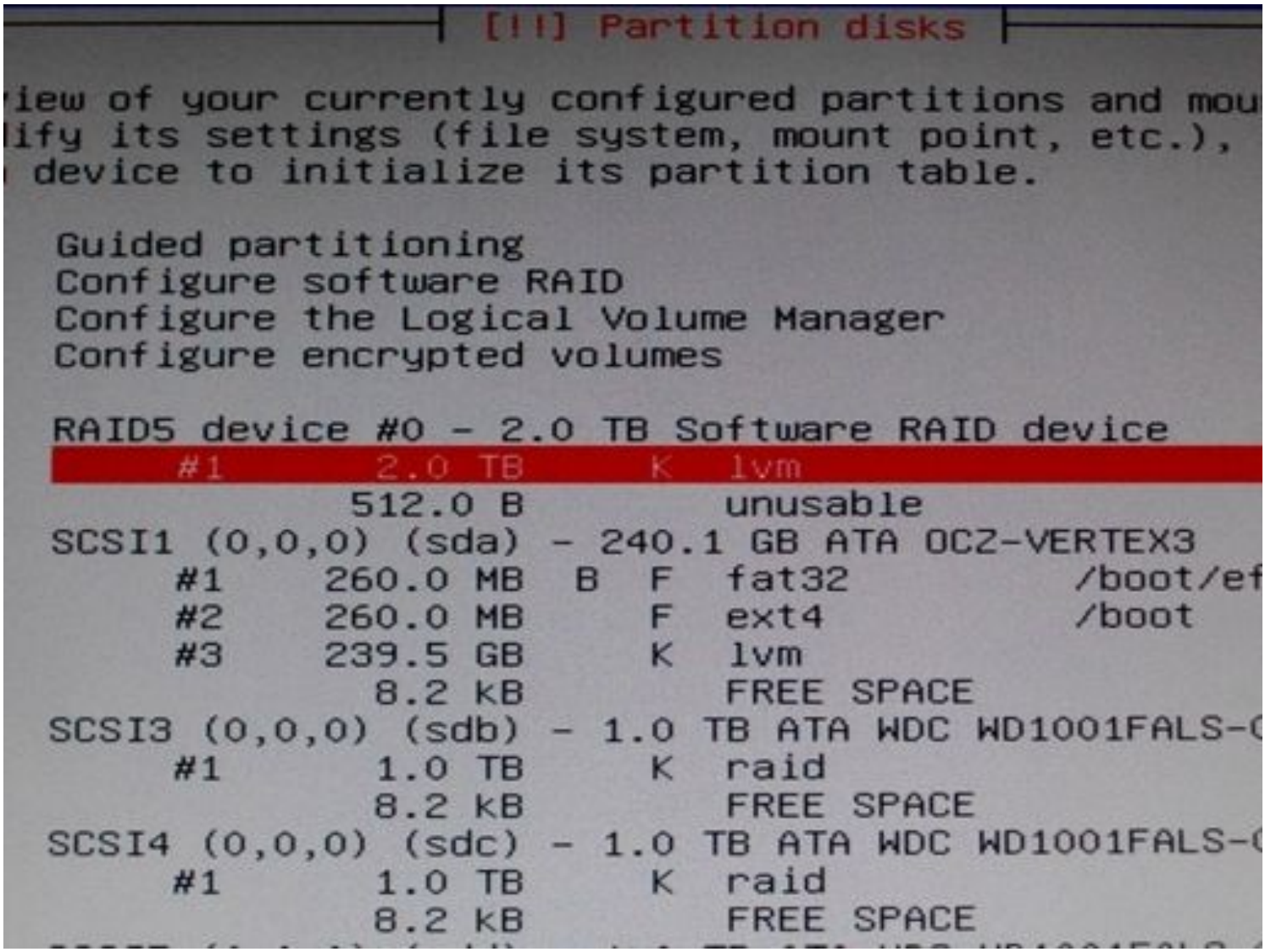

[!!] Partition disks
The RAID5 array will consist of both active and spare devi
those used, while the spare devices will only be used if o
devices fail. A minimum of 3 active devices is required.
NOTE: this setting cannot be changed later.
Number of active devices for the RAID5 array:
3
<Go Back>
Tab> moves; <Space> selects; <Enter> activates buttons



Now we get to select which disks are part of the RAID from all available partitions:

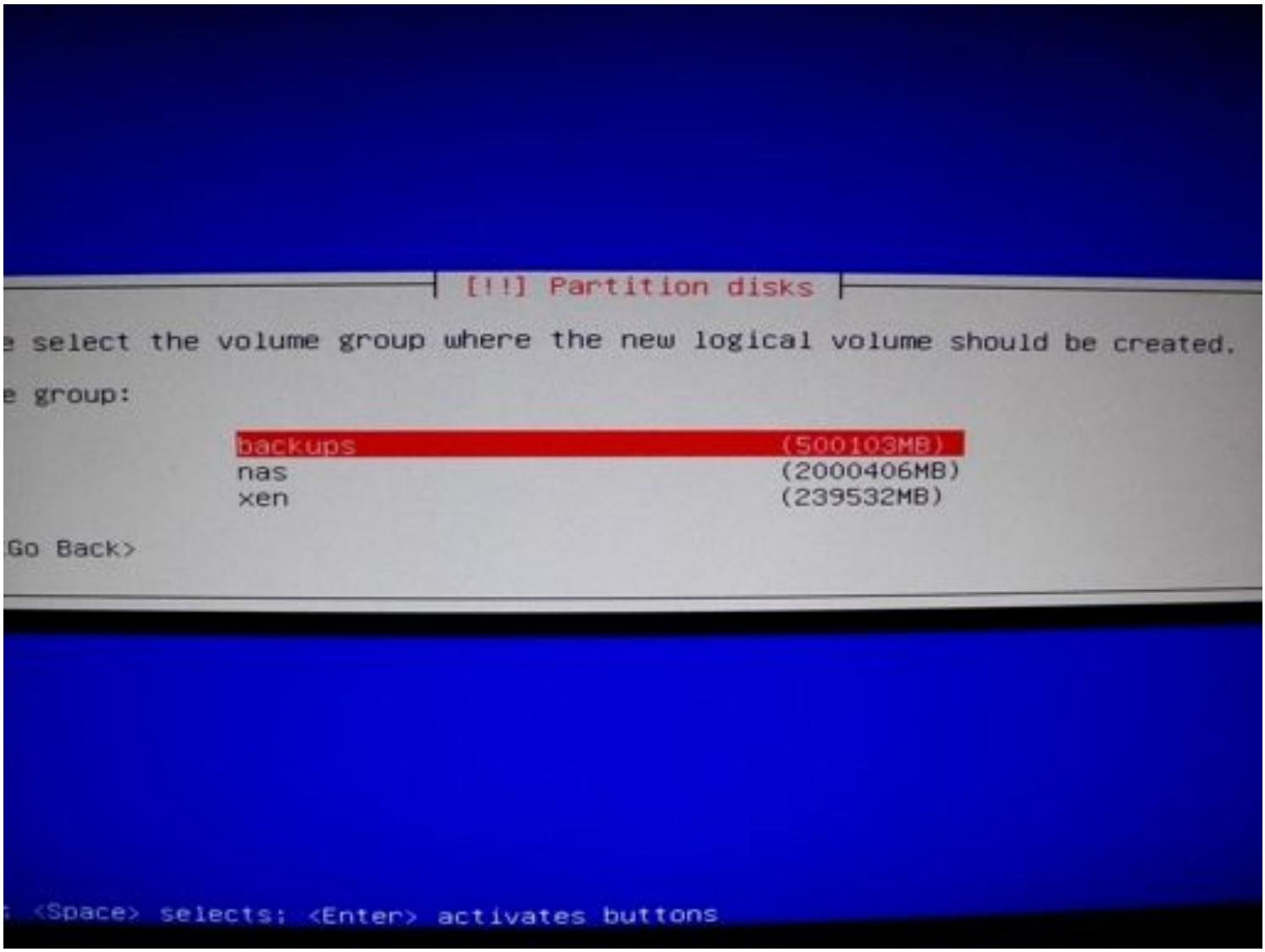


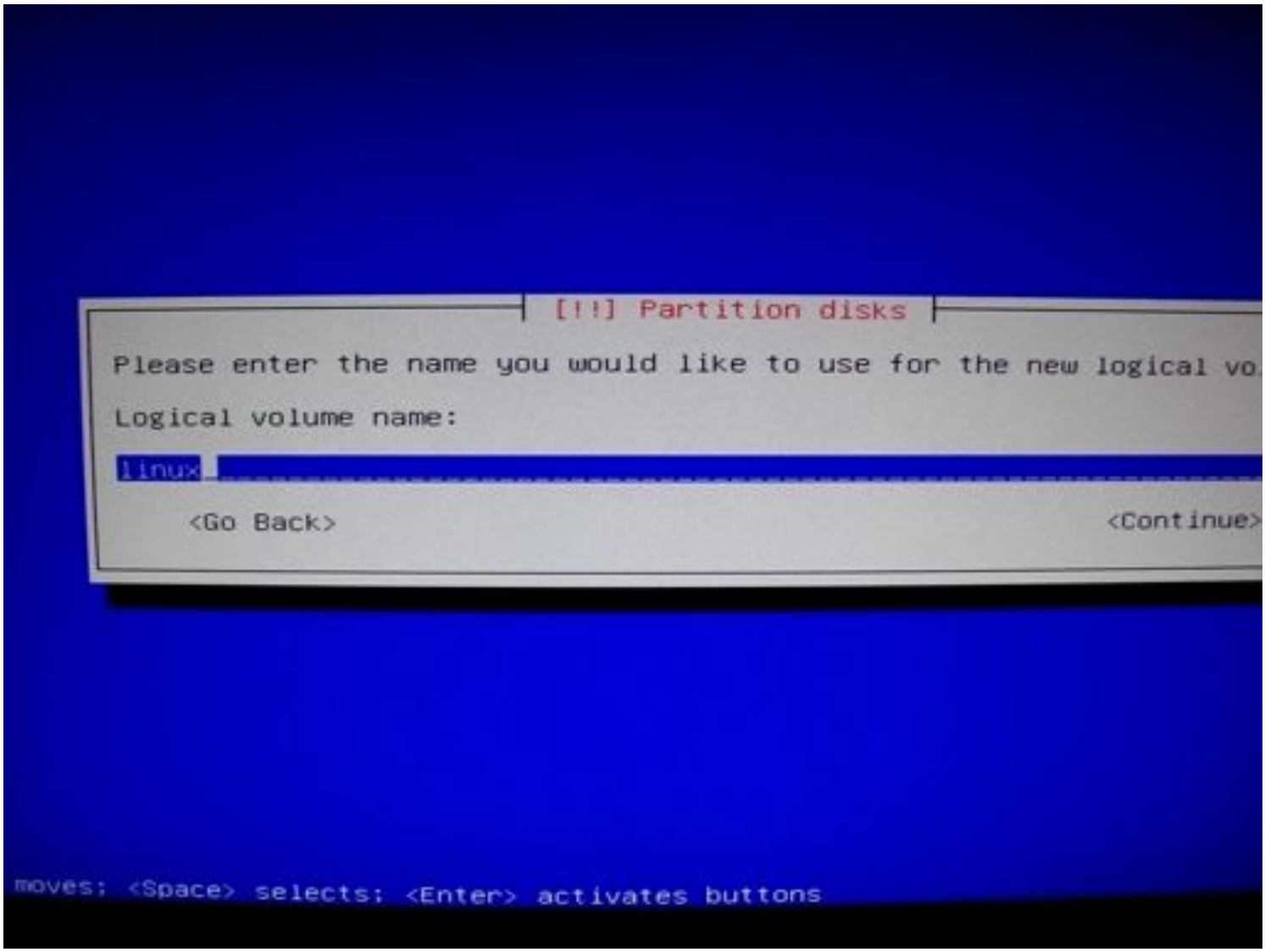
Once that is done we partition our RAID, in this case I made a 2 TB LVM partition:



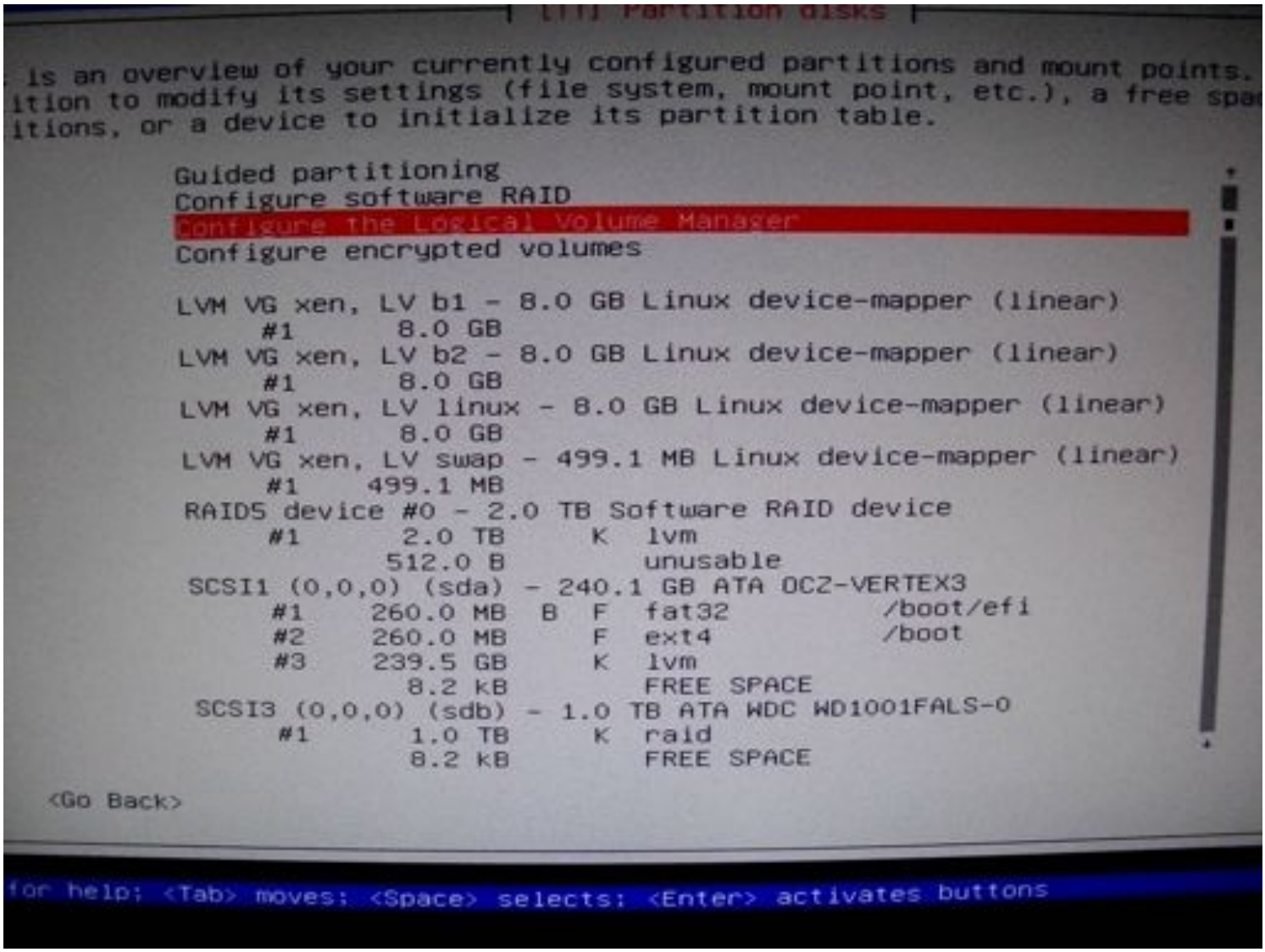
For the LVM configuration that follows please note that my final disk sizes changed, and the images do not reflect this. Be sure to read as well as use the images as a guide.

Next we want to setup our LVM's, it will ask us again to confirm any disk changes. We only have one option, to create a volume group, when selected you can specify a name, and select LVM partitions. For our purposes we only want one partition per group, but as an alternative to using striped RAID you could just use LVM's. Once we have our groups, we can select to create a logical volume and we will be asked which group, a name for the volume, and the size:

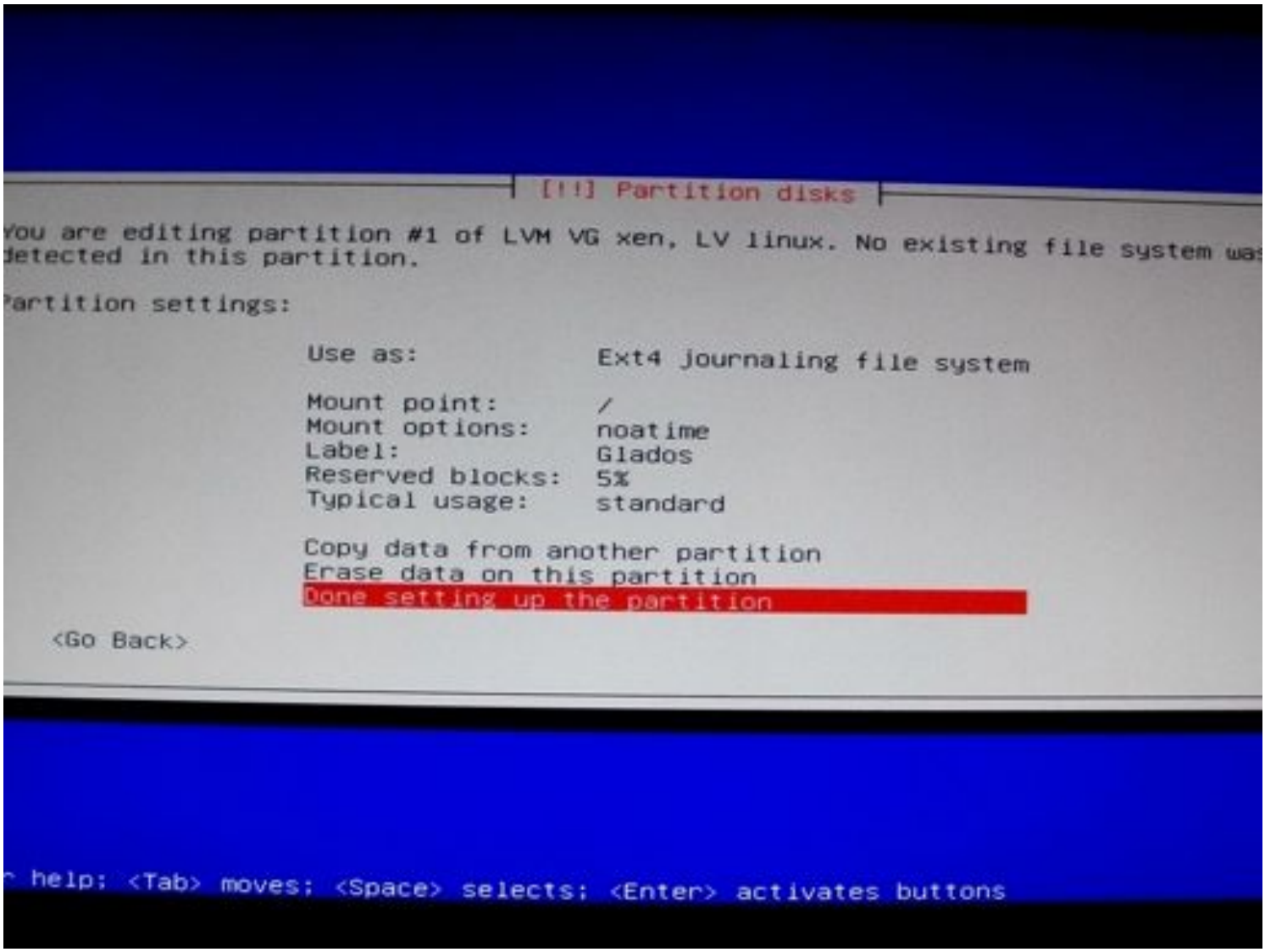

[!!] Partition disks
select the volume group where the new logical volume should be created.
group:
backups (500103MB)
nas (2000406MB)
xen (239532MB)
Go Back>
<Space> selects; <Enter> activates buttons



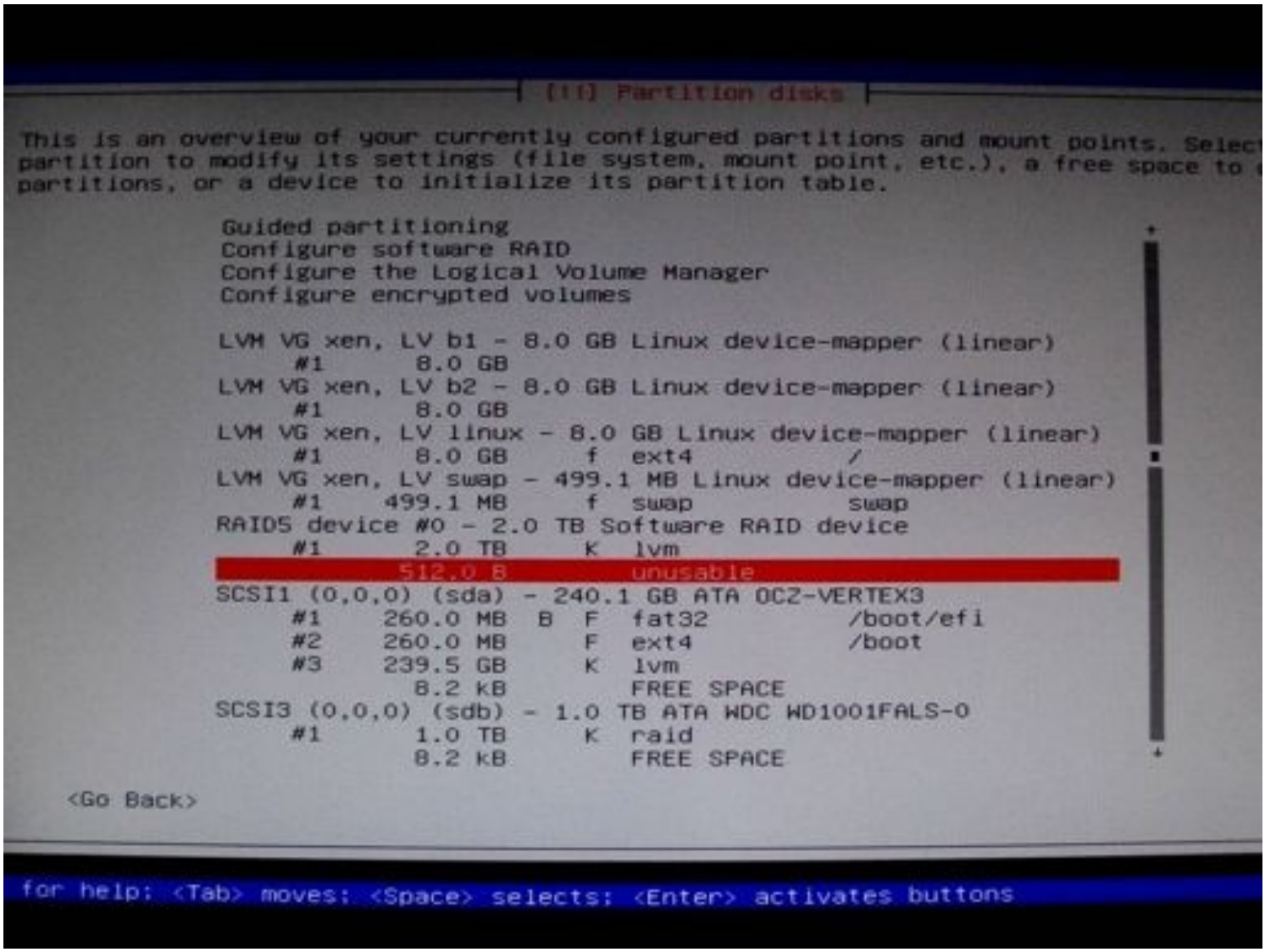
We need three volumes for our initial setup. The first for the root partition, I set 6GB of space. Second I set 500 MB of space for a swap partition. Finally I would recommend 18GB of space for a User partition, which can store our compiled data and installation files for our virtual machines. Once you select done, you should see a similar display of available partitions:



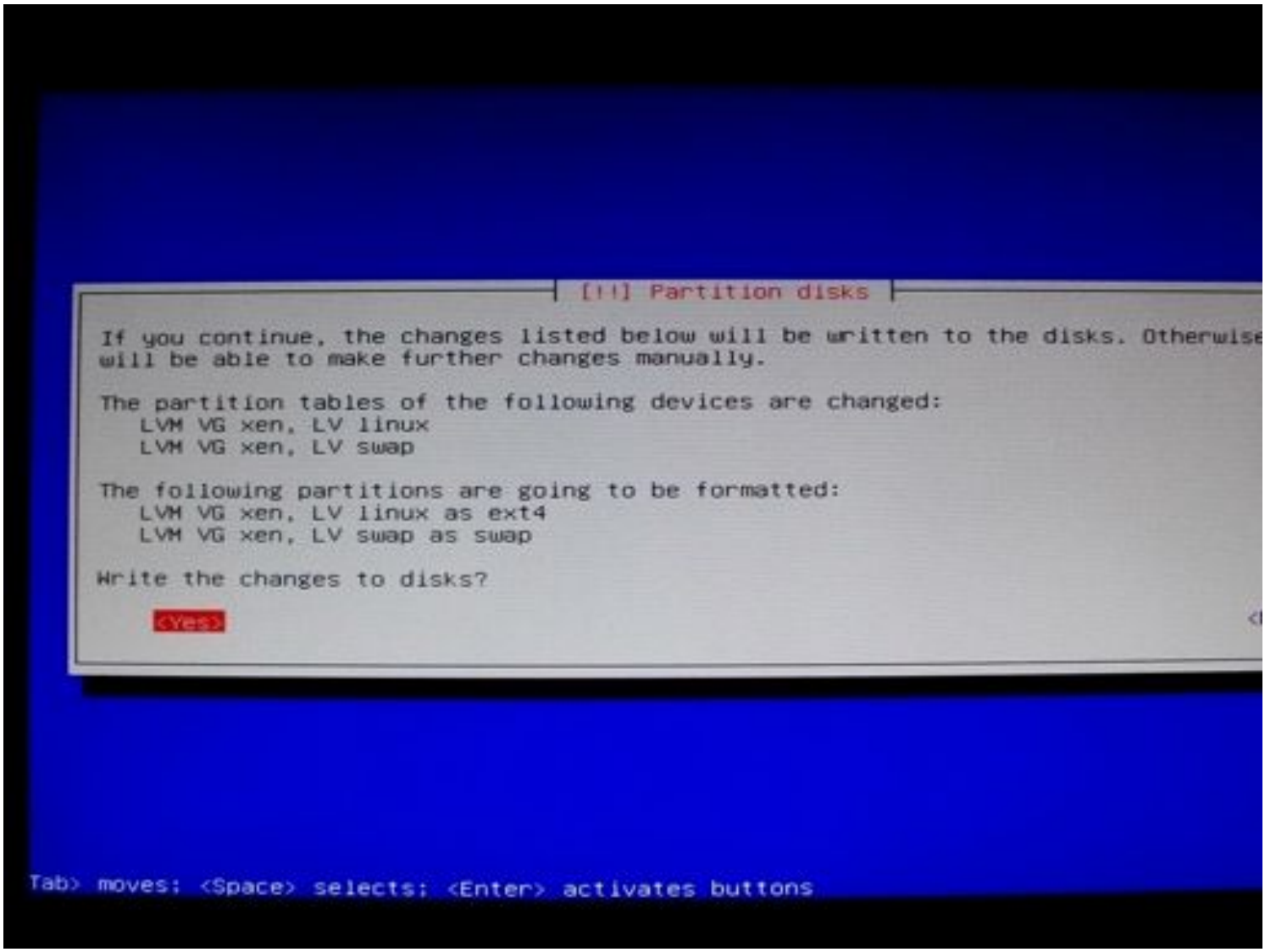
For our home partition we set ext4 and mount point to "/".



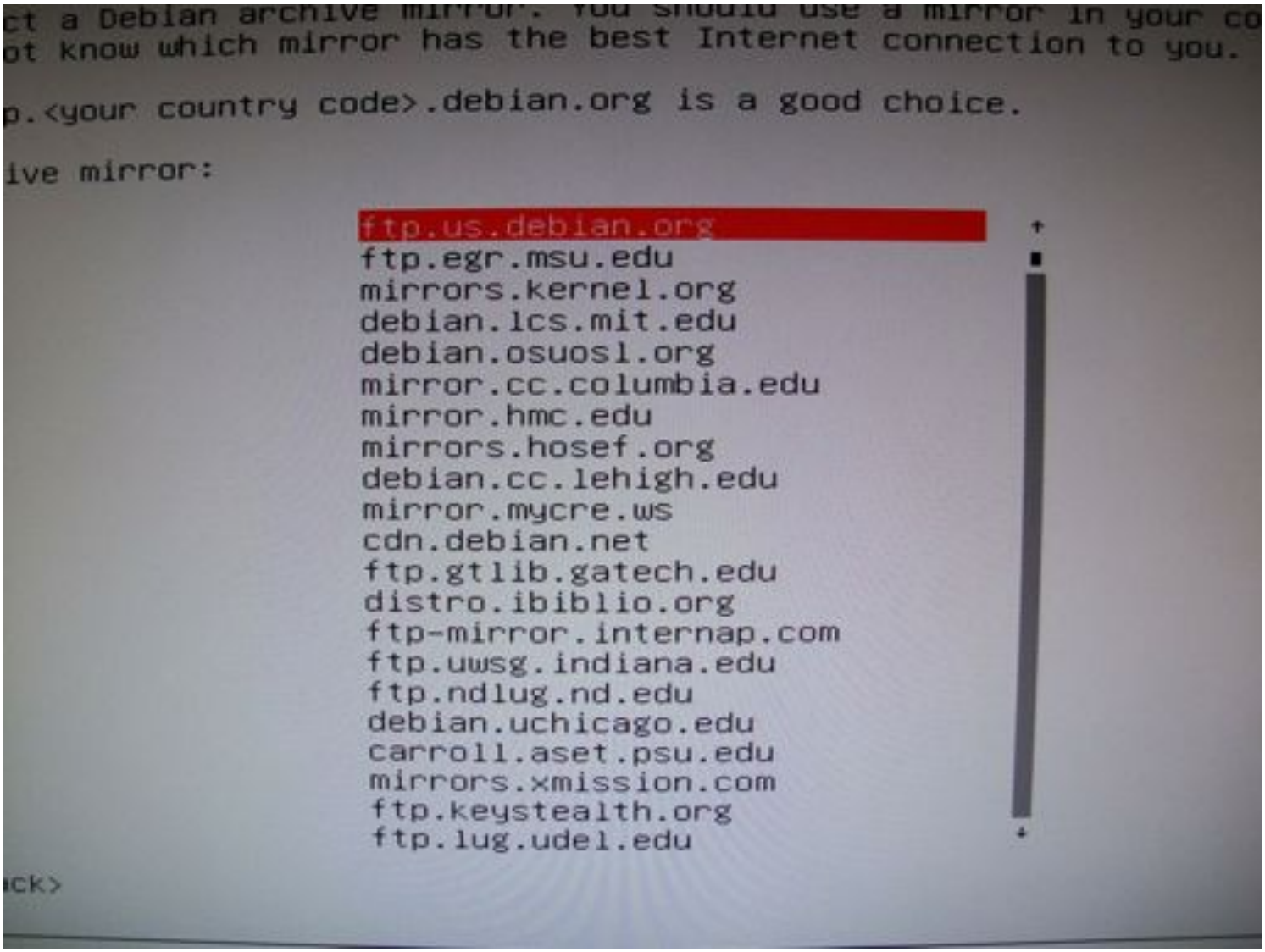
I also set the 500MB partition as swap space:



The 18GB Partition I set to mount at /home, and a filesystem of ext4. At the bottom you can confirm the configuration and move forward:

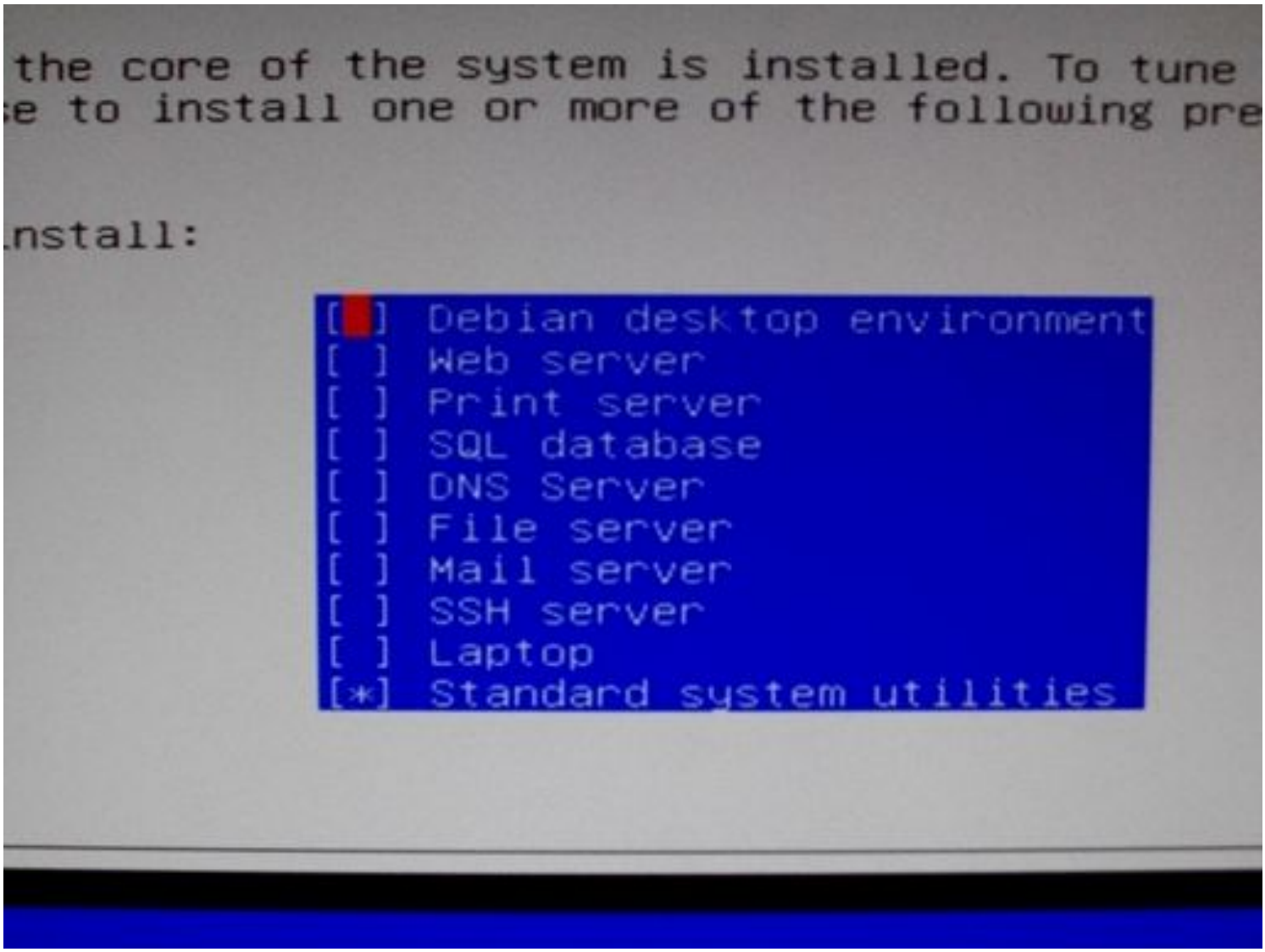


As before we will just confirm each menu as we go through, eventually you'll get to this one:



Any of the mirrors are fine, if it fails you can try another or just try the same one a second time.

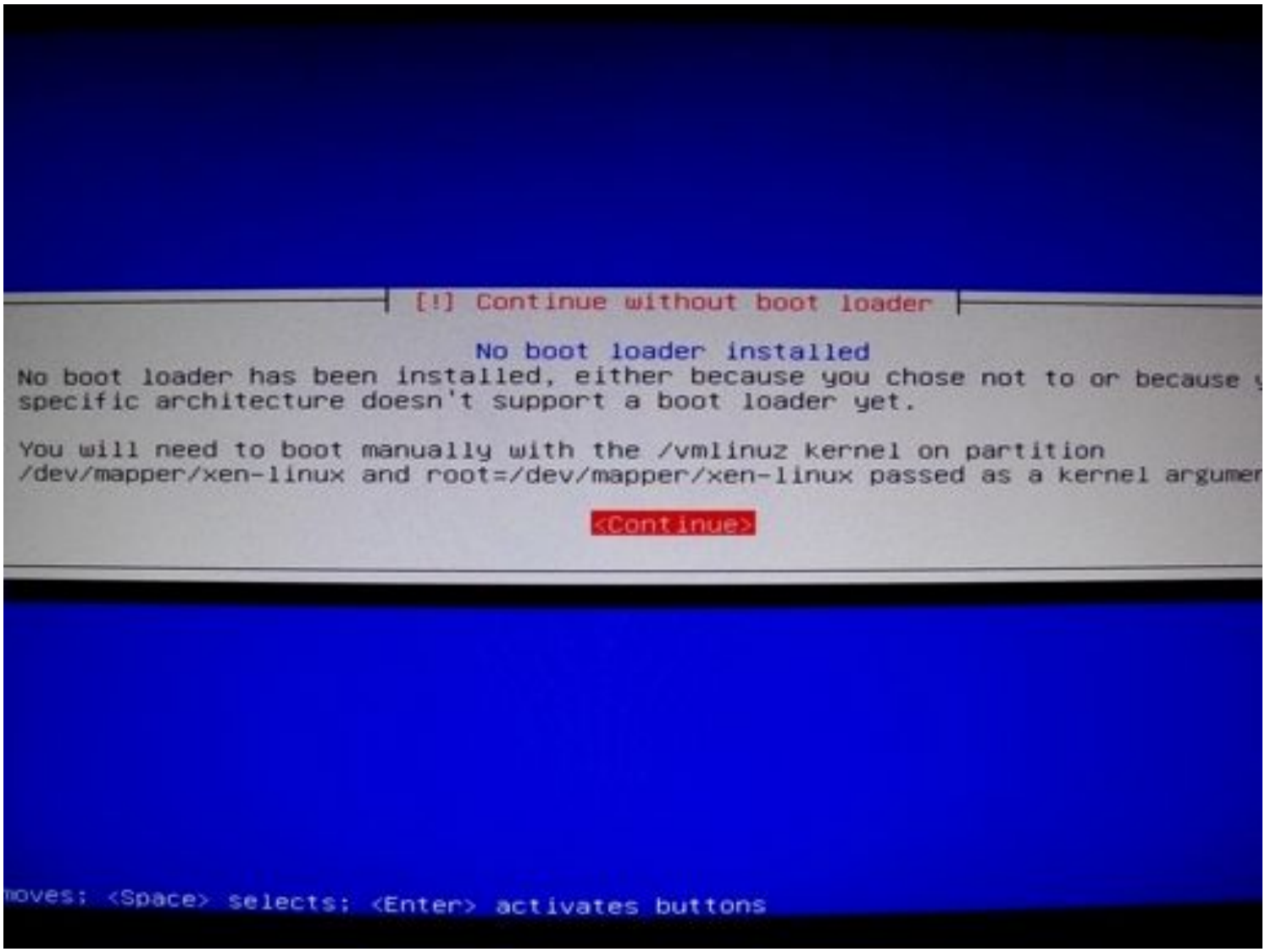
For a bare-bones installation you would usually de-select the graphical user interface from this screen:



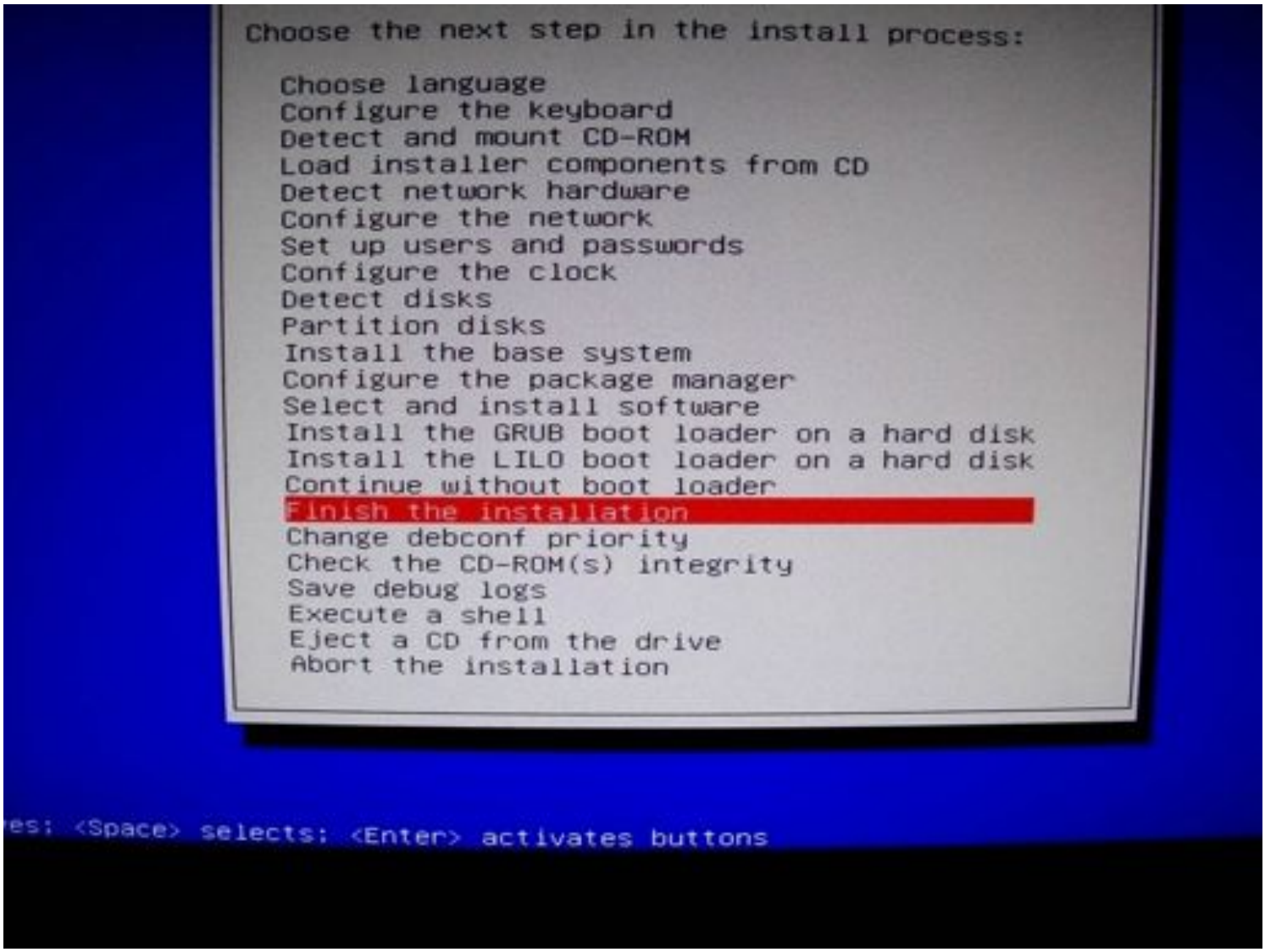
If you do not have a second machine to work from, or want a GUI for Dom0, then feel free to select it.

Important! The installer automatically defaults to installing the grub bootloader, and if we are trying to install the efi grub bootloader without making a mess of our disk we want to go down two to the "Continue without a bootloader" option:

You will be given a very important message, this information is used for the manual boot process so be sure you can remember it, or write it down:

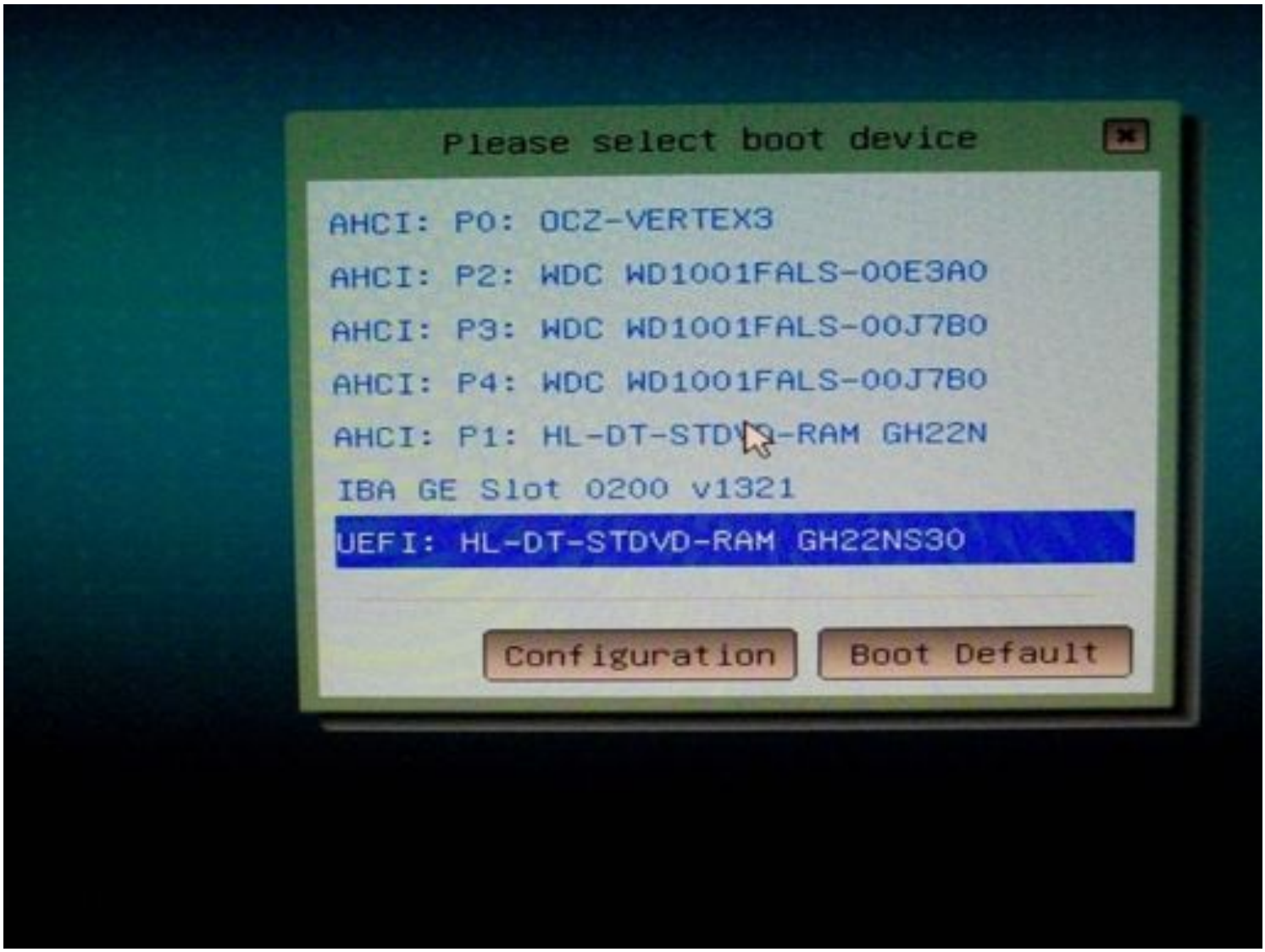


Now we can finish the installation, when it pops the install disk out pop the Ubuntu Live CD in and proceed onto the next step to Manually Boot Linux!

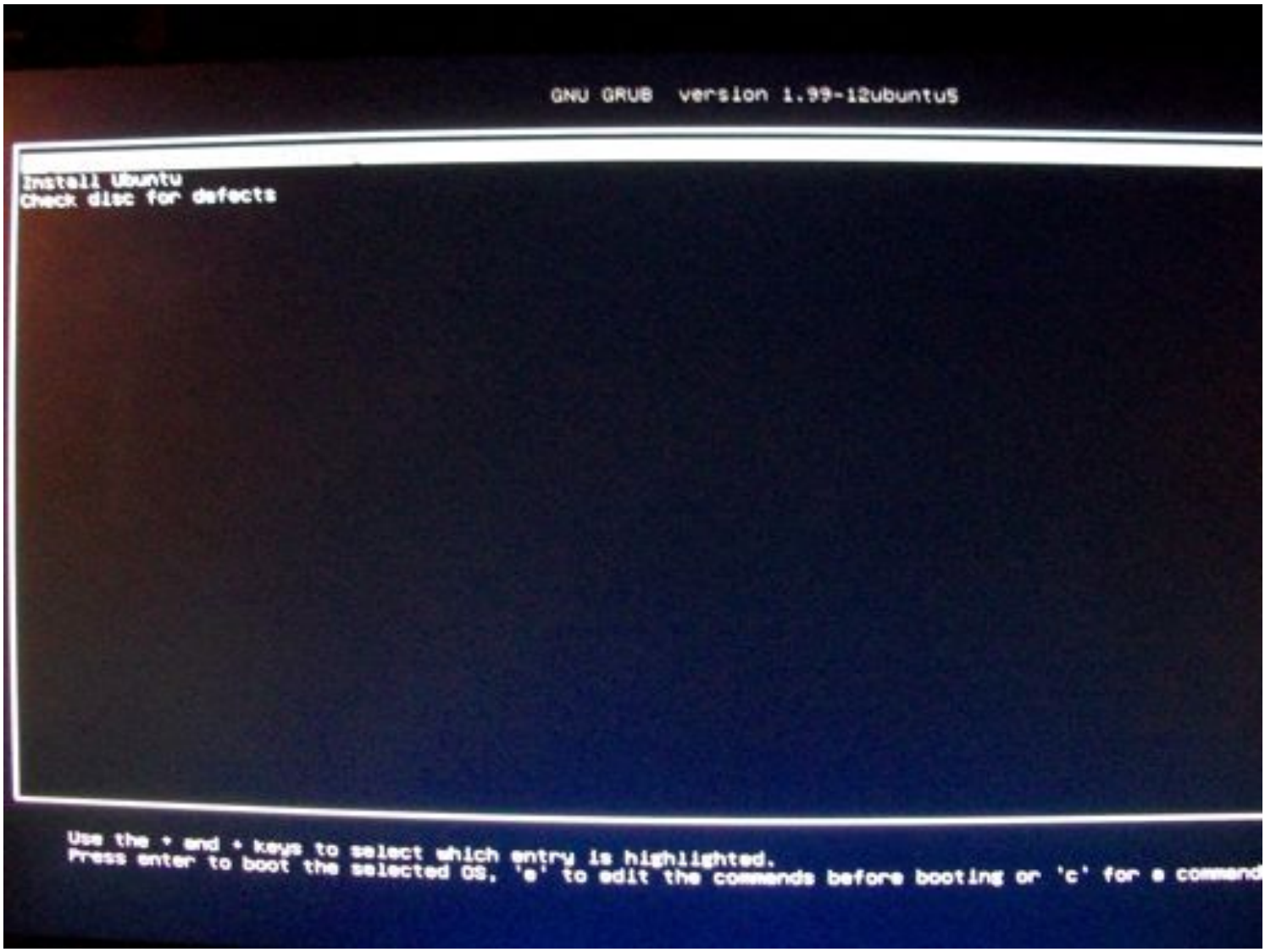


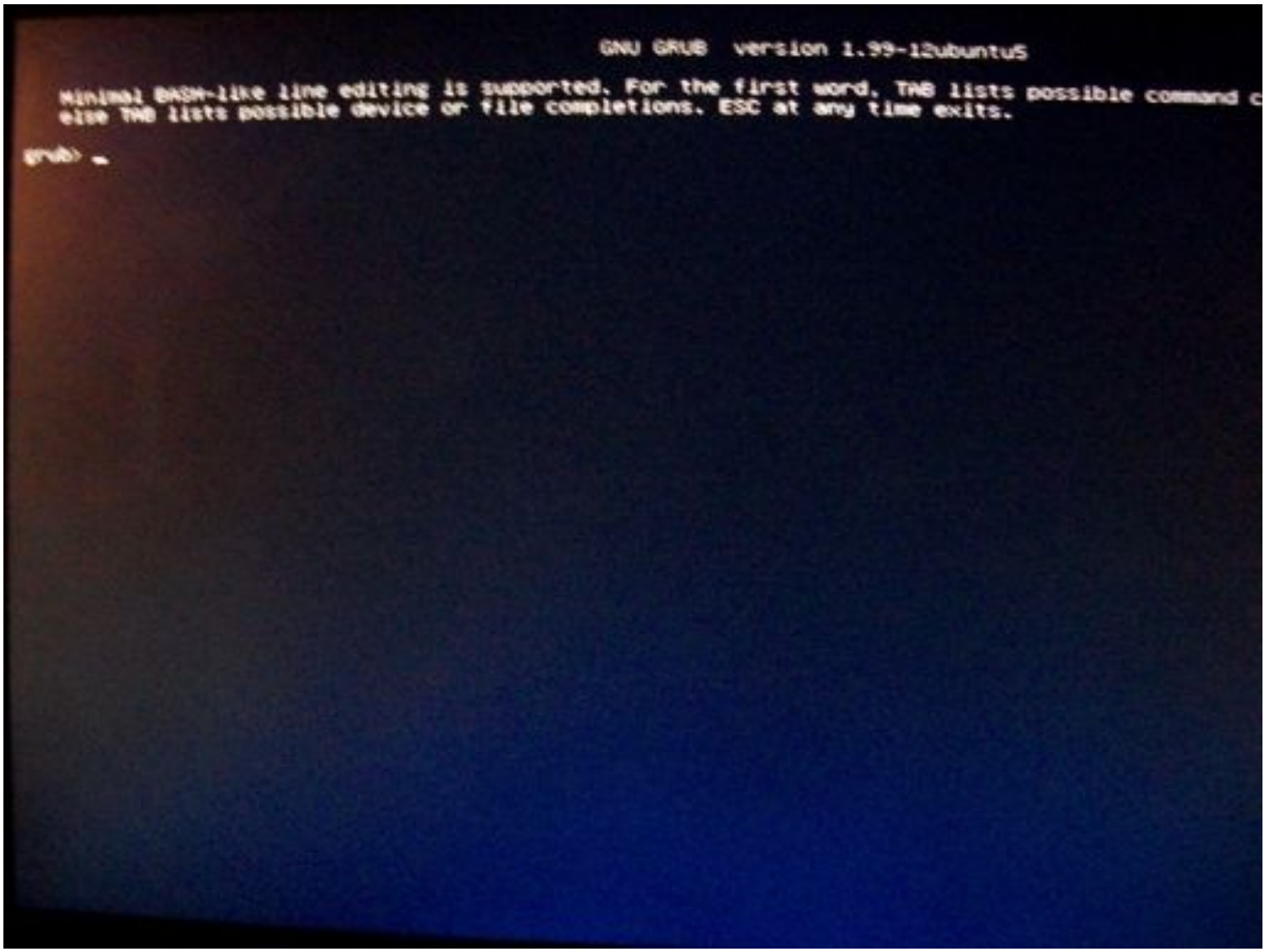
Manually booting Wheezy

Once the system begins rebooting the very first thing we want to do is load the boot option menu for our system and specifically select the UEFI boot option for the Ubuntu Live CD:

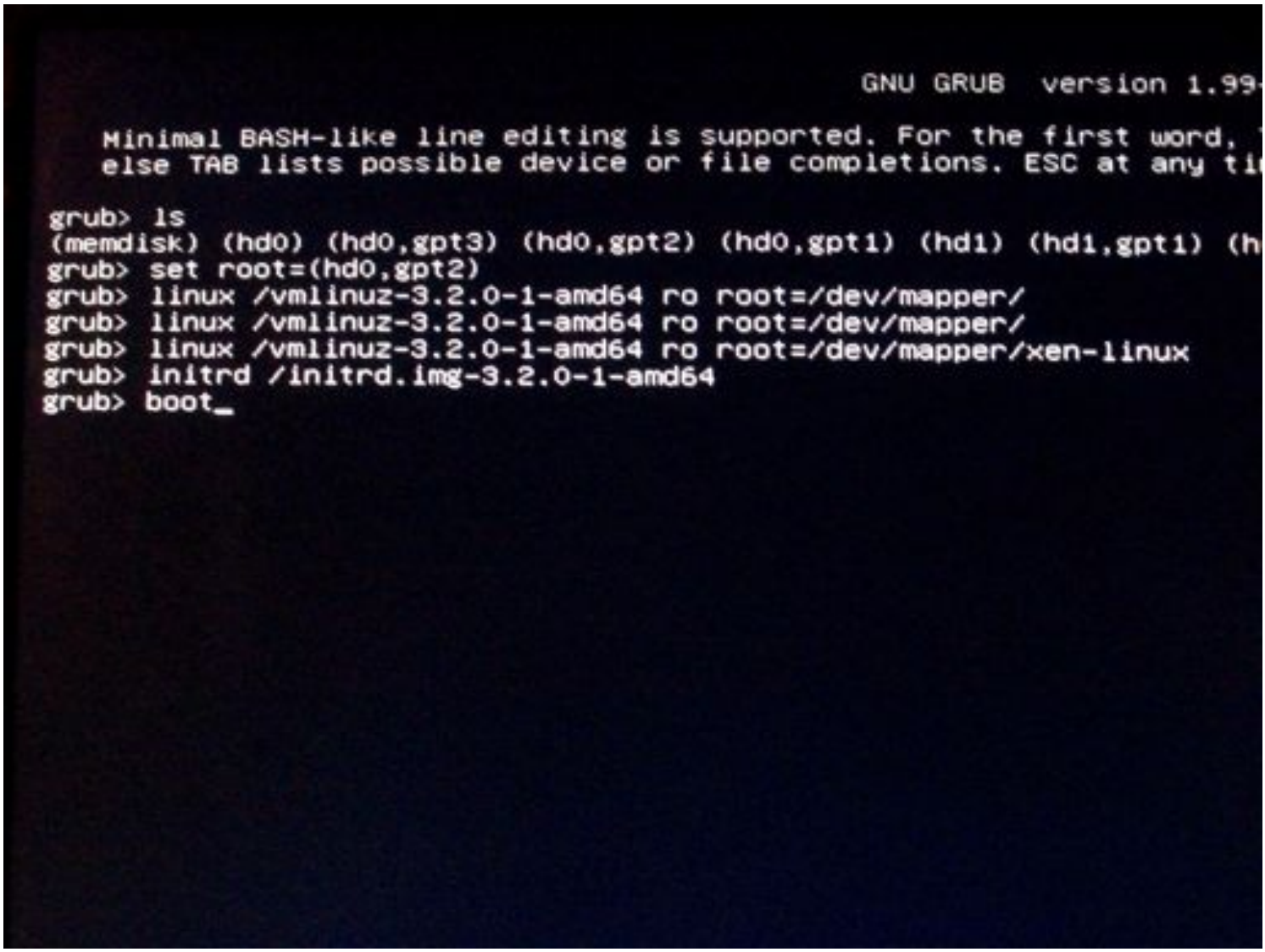


This will give us a very different looking black and white menu, from here we can press "c" on the keyboard to access the grub emergency system:


GNU GRUB version 1.99-12ubuntu5
Install Ubuntu
Check disc for defects
Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands before booting or 'c' for a command



I am going to jump ahead and show you want the commands you need to enter look like, then explain them:




```
set root=(hd0,gpt2)
linux /vmlinux ro root=/dev/mapper/???
initrd /initrd
boot
```

Manually booting seems confusing at first, but it is actually quite simple. The grub menu is a mini terminal with specific commands, ls for example will list your hard drives and partitions.

Setting the "root" variable tells grub which partition to look for files in. The files it needs are the kernel and friends, which means you want to point it to the partition you made "/boot". If you were following my guide that should be the middle partition, or gpt2.

The second command specifies the kernel we are using, and passes the base directory to the kernel, in this case our LVM partition. Grub cannot read the LVM partitions which is why we separated the "/boot" directory.

You will notice that the grub system has auto-complete, hitting tab will fill in the blanks if you don't know the full kernel name. The parameters we pass are "ro" and "root=", and if you recall from the previous step we got a message when we opted to not install a bootloader. It tells us the LVM partition name to use and assign that to the root flag.

Just like with the linux kernel, we set the initrd file which should also be in the /boot partition.

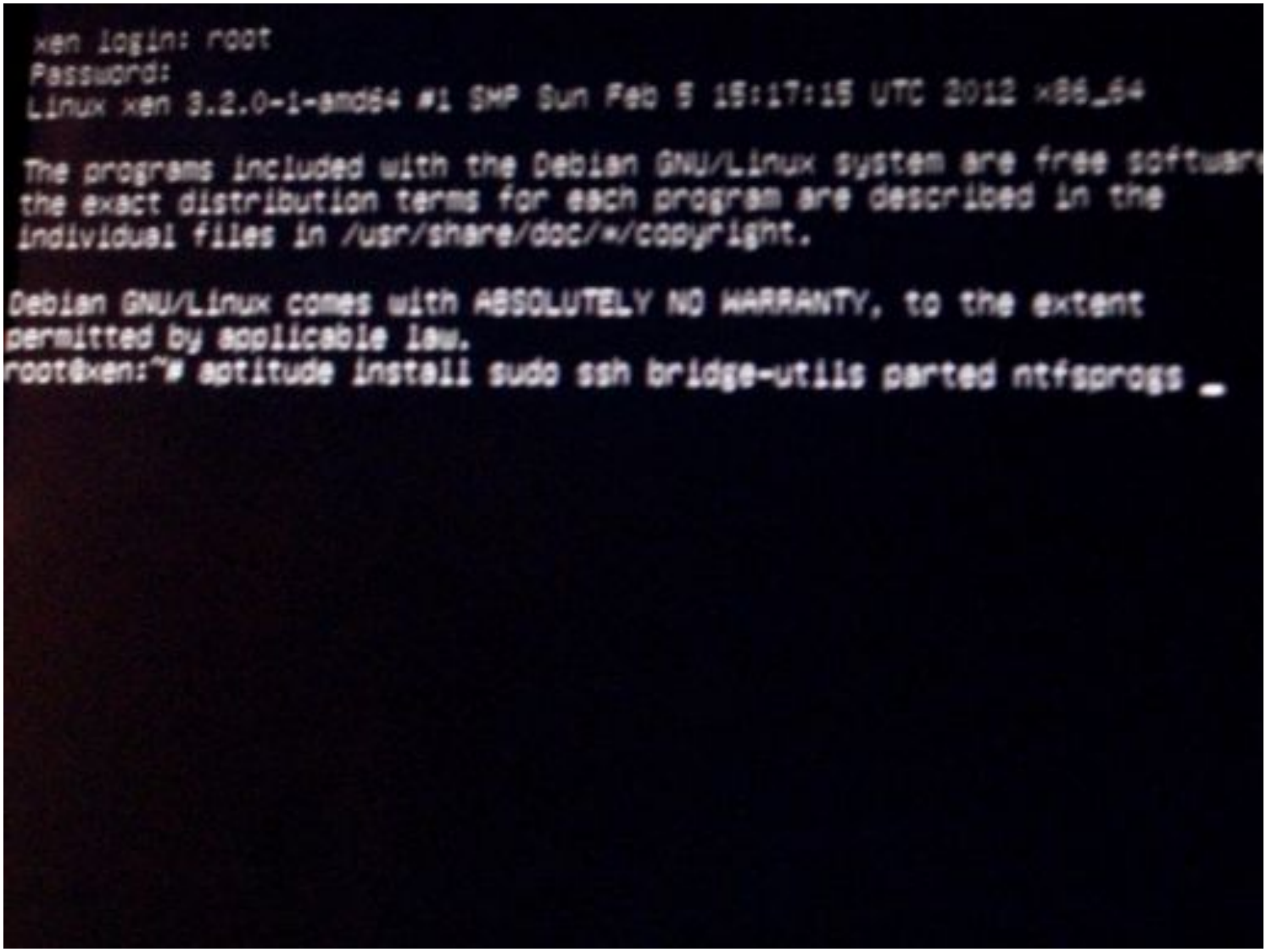
Finally, when you type "boot" and hit enter, it uses the previous settings to load linux!

A quick note, if you chose not to use LVM for your base system, and /boot is a part of root ("/"), then you can select the root partition and just use longer paths to the kernel and initrd (/boot/ instead of /).

If all went as planned you will be greeted shortly by your login prompt. Don't be alarmed is goofy glitches occur with output or the video, emergency grub isn't perfect. From here we will want to install EFI and some basic utilities.

Installing EFI & Basic Linux Configuration

The default Wheezy installation will not have sudo setup so I recommend logging in as root first, and running this to quickly install some utilities onto the system:

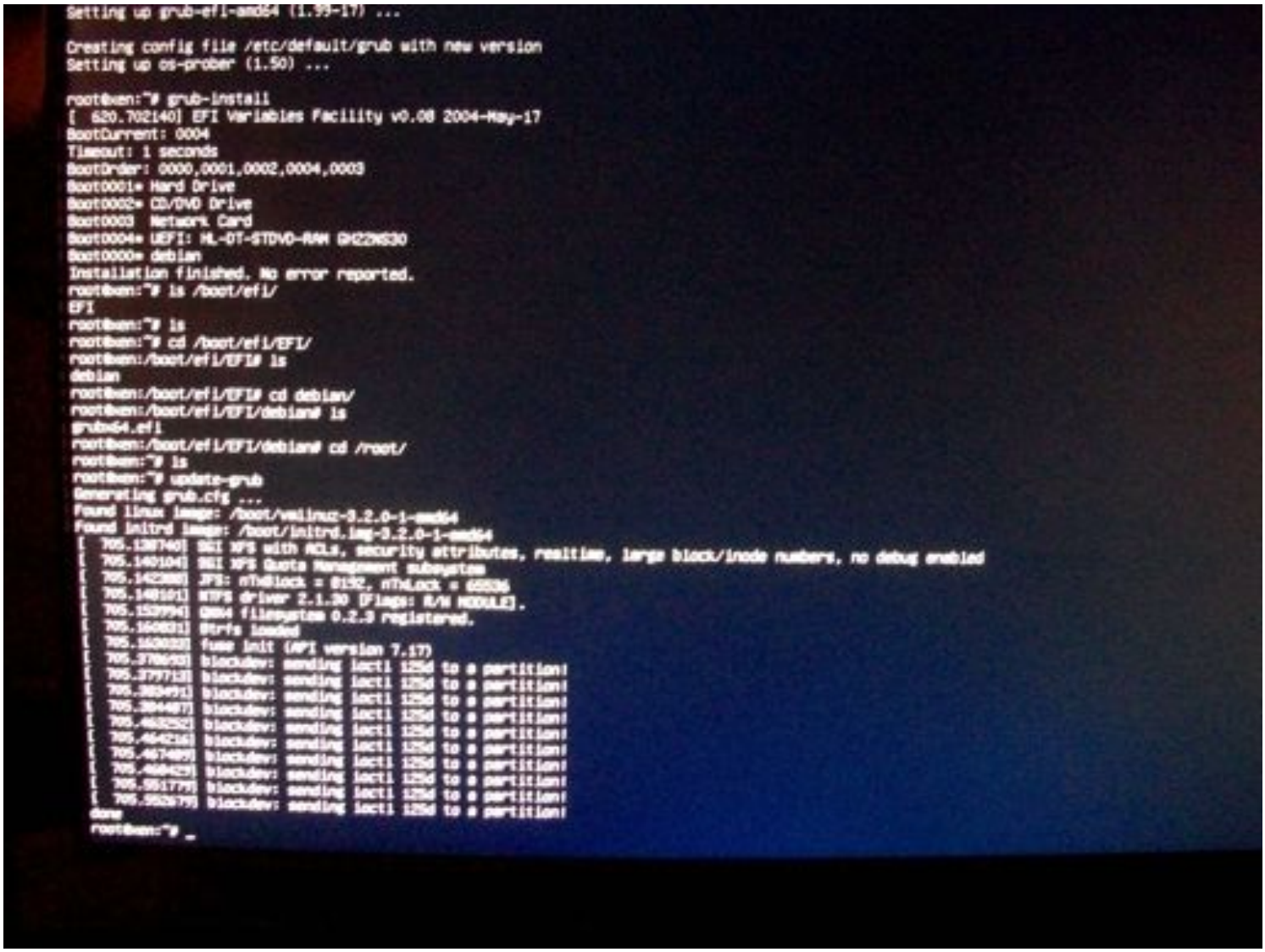



```
aptitude install sudo ssh bridge-utils parted ntfsprogs grub-efi-amd64
```

If you want to install all the utilities we are going to need, feel free to do so now as well:

```
sudo aptitude install grub-efi-amd64 sudo ssh bridge-utils parted ntfsprogs bzip2
build-essential libncurses-dev kernel-package fakeroot python-dev uuid-dev libglib2.0-dev
libyajl-dev bcc gcc-multilib iasl libpci-dev mercurial
```

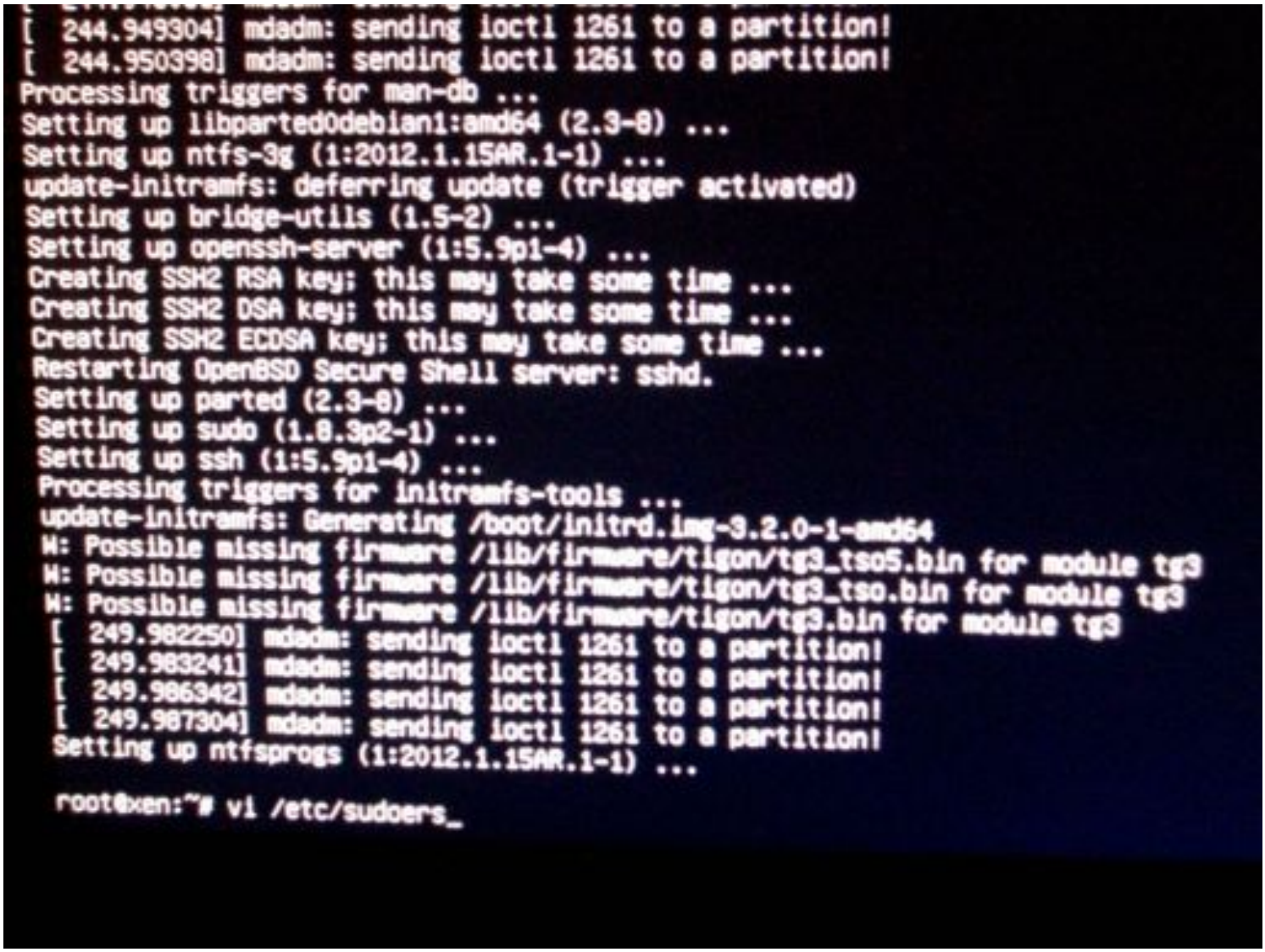
Once these basic utilities are installed we want to run grub-install, and update-grub:

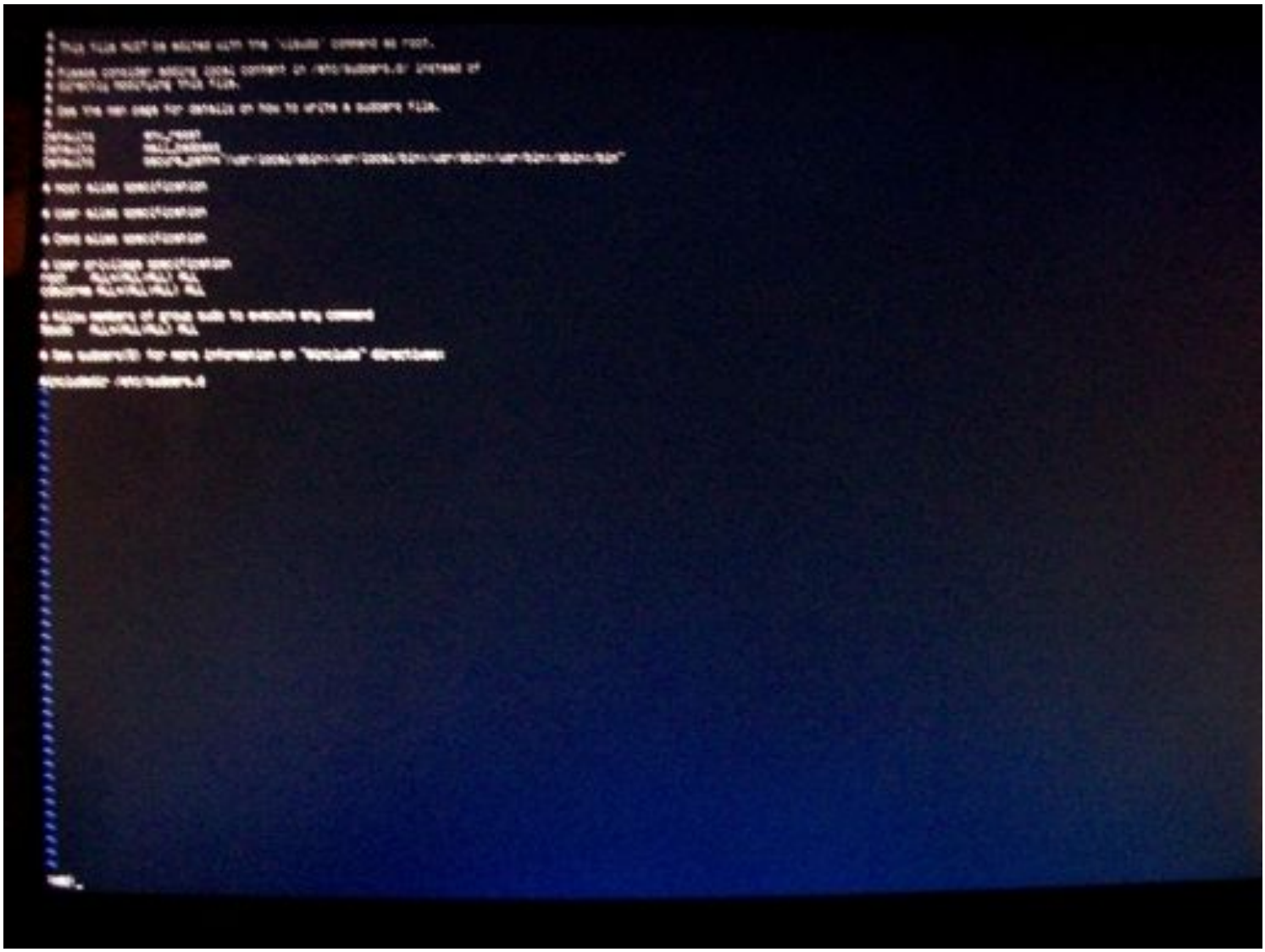



```
grub-install && update-grub
```

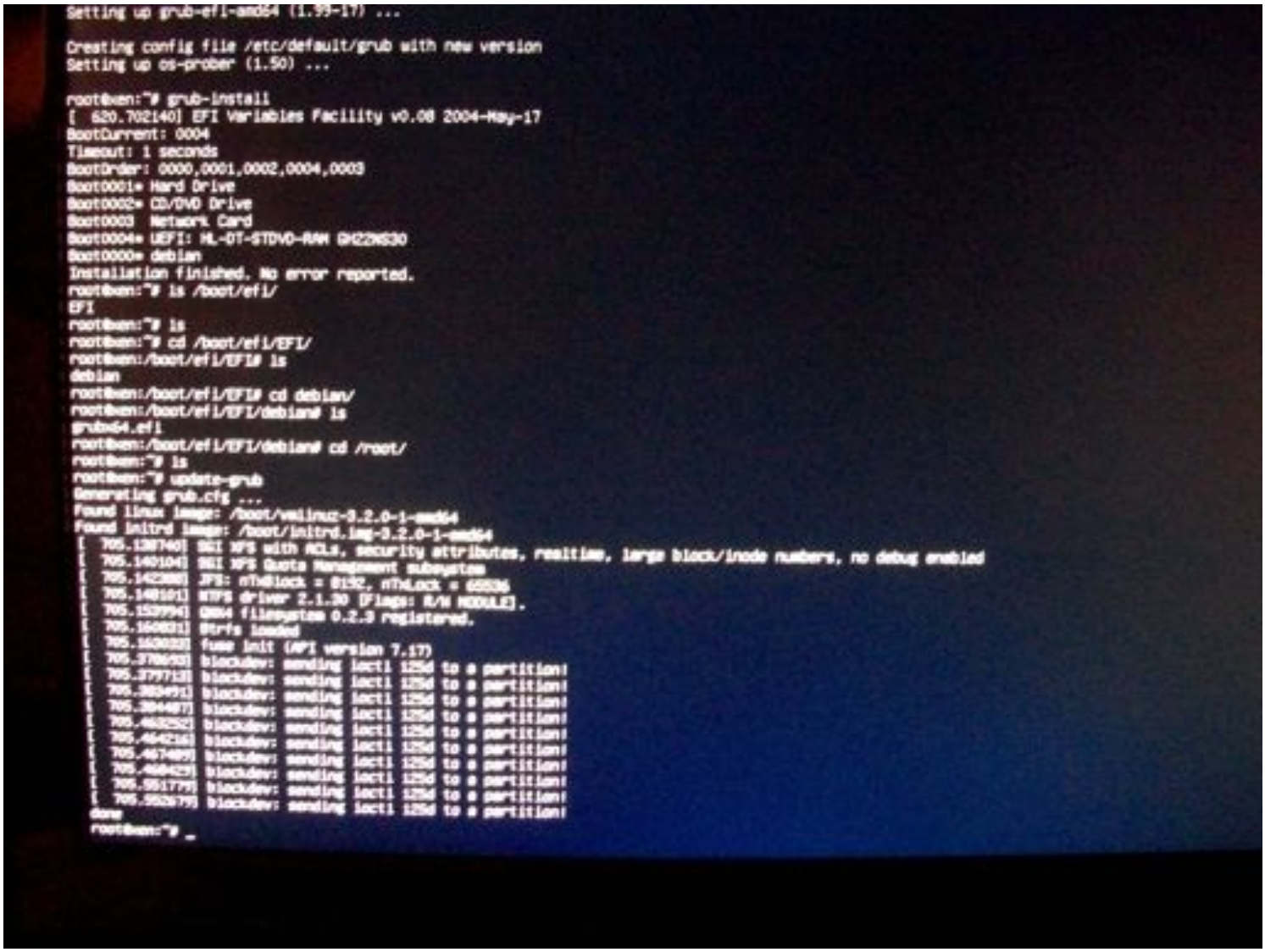
The install will create an EFI/debian folder, containing grubx64.efi, prepare the basic configuration, and if you used the UEFI Live CD to boot, it should run a command to inform your UEFI BIOS of the new boot partition. If you didn't use the UEFI Live CD then you may have to go through your UEFI Bios to boot and reinstall grub as an extra step.

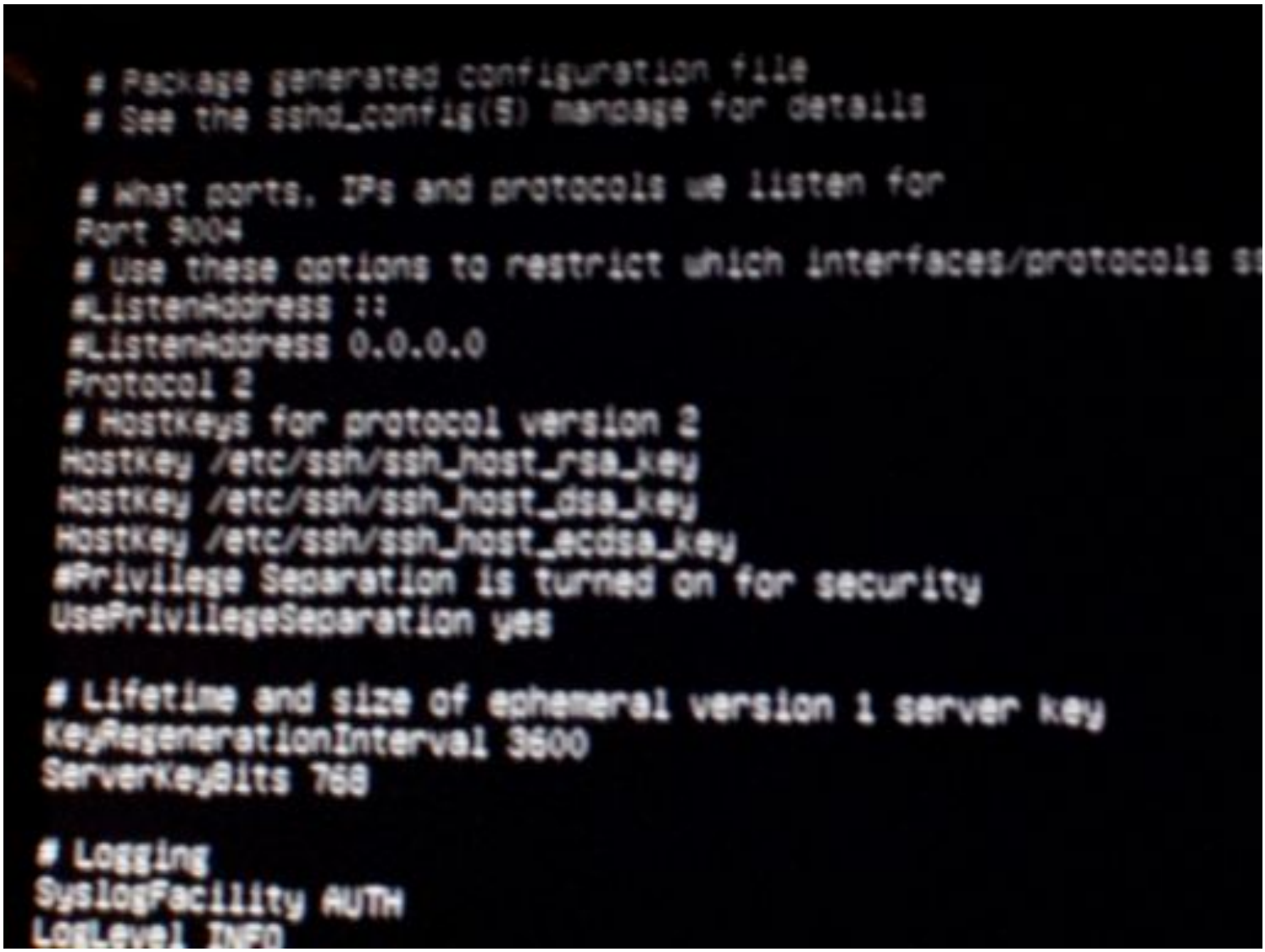
You can test rebooting now if you want, but I like to make some modifications first. I add my user to the /etc/sudoers file, since this file is special you will have to use "wq!" from vi to write and close:


[244.949304] mdadm: sending ioctl 1261 to a partition!
[244.950398] mdadm: sending ioctl 1261 to a partition!
Processing triggers for man-db ...
Setting up libparted0debian1:amd64 (2.3-8) ...
Setting up ntfs-3g (1:2012.1.15AR.1-1) ...
update-initramfs: deferring update (trigger activated)
Setting up bridge-utils (1.5-2) ...
Setting up openssh-server (1:5.9p1-4) ...
Creating SSH2 RSA key; this may take some time ...
Creating SSH2 DSA key; this may take some time ...
Creating SSH2 ECDSA key; this may take some time ...
Restarting OpenBSD Secure Shell server: sshd.
Setting up parted (2.3-8) ...
Setting up sudo (1.8.3p2-1) ...
Setting up ssh (1:5.9p1-4) ...
Processing triggers for initramfs-tools ...
update-initramfs: Generating /boot/initrd.img-3.2.0-1-amd64
W: Possible missing firmware /lib/firmware/tigon/tg3_tso5.bin for module tg3
W: Possible missing firmware /lib/firmware/tigon/tg3_tso.bin for module tg3
W: Possible missing firmware /lib/firmware/tigon/tg3.bin for module tg3
[249.982250] mdadm: sending ioctl 1261 to a partition!
[249.983241] mdadm: sending ioctl 1261 to a partition!
[249.986342] mdadm: sending ioctl 1261 to a partition!
[249.987304] mdadm: sending ioctl 1261 to a partition!
Setting up ntfsprogs (1:2012.1.15AR.1-1) ...
root@xen:~# vi /etc/sudoers_

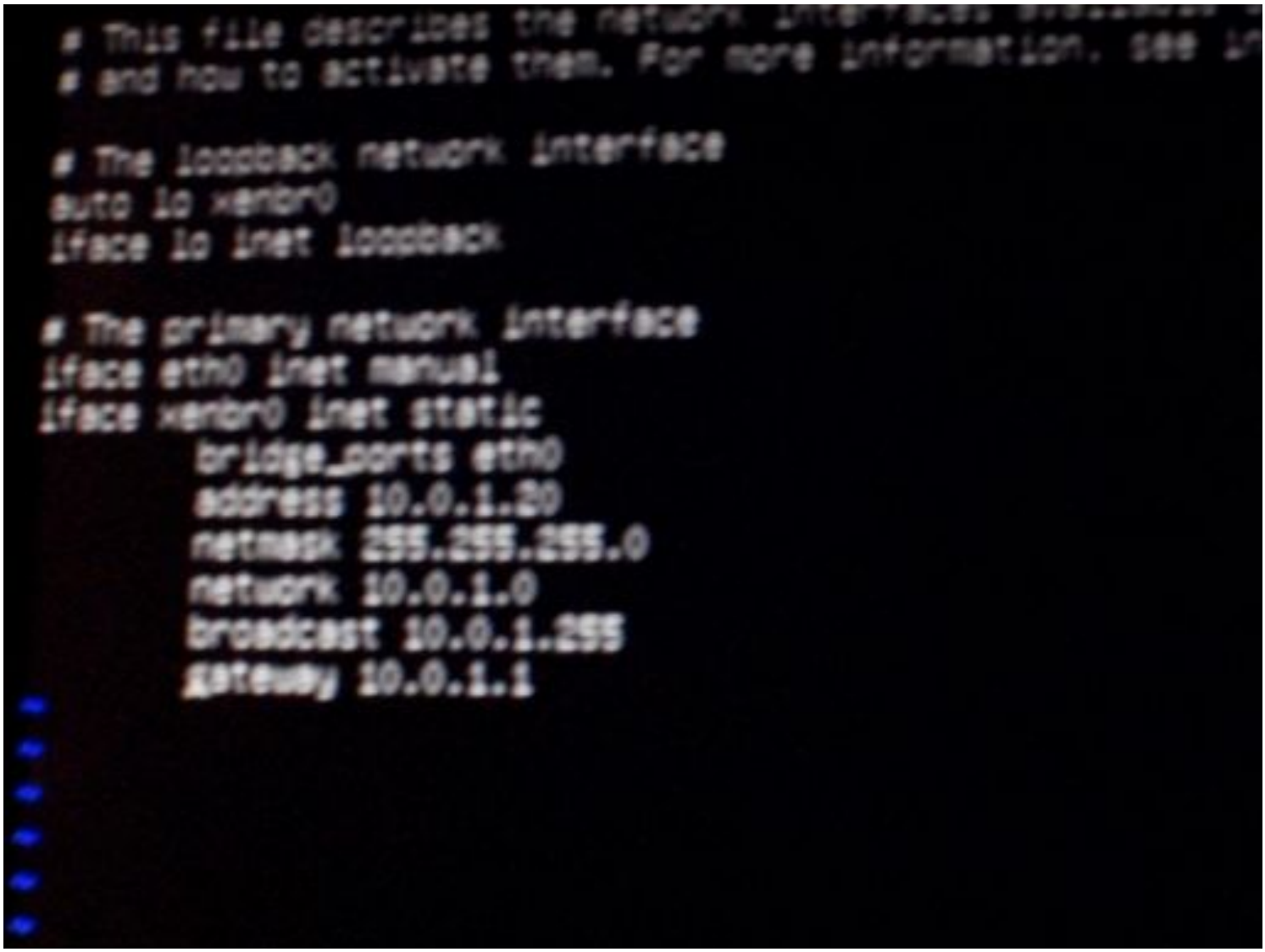


Next I change the ssh port in /etc/ssh/sshd_config, this step is 100% optional, but I find it helpful for security among other things. The port is the fourth option down in the file, super easy to find.


Setting up grub-efi-amd64 (1.99-17) ...
Creating config file /etc/default/grub with new version
Setting up os-prober (1.50) ...
root@ven:~# grub-install
[620.702140] EFI Variables Facility v0.08 2004-May-17
BootCurrent: 0004
Timeout: 1 seconds
BootOrder: 0000,0001,0002,0004,0003
Boot0001* Hard Drive
Boot0002* CD/DVD Drive
Boot0003 Network Card
Boot0004* UEFI: HL-DT-STDVD-RAM GH22NS30
Boot0000* debian
Installation finished. No error reported.
root@ven:~# ls /boot/efi/
EFI
root@ven:~# ls
root@ven:~# cd /boot/efi/EFI/
root@ven:/boot/efi/EFI# ls
debian
root@ven:/boot/efi/EFI# cd debian/
root@ven:/boot/efi/EFI/debian# ls
grubx64.efi
root@ven:/boot/efi/EFI/debian# cd /root/
root@ven:~# ls
root@ven:~# update-grub
Generating grub.cfg ...
Found linux image: /boot/vmlinuz-3.2.0-1-amd64
Found initrd image: /boot/initrd.img-3.2.0-1-amd64
[705.138740] SGI XFS with ACLs, security attributes, realtime, large block/inode numbers, no debug enabled
[705.140104] SGI XFS Quota Management subsystem
[705.145288] JFS: nTxBlock = 8192, nTxLock = 65536
[705.148161] NTFS driver 2.1.30 [Flags: R/W MODULE].
[705.153794] QNX4 filesystem 0.2.3 registered.
[705.160831] Btrfs loaded
[705.163033] fuse init (API version 7.17)
[705.370693] blockdev: sending ioctl 125d to a partition!
[705.377713] blockdev: sending ioctl 125d to a partition!
[705.383491] blockdev: sending ioctl 125d to a partition!
[705.384487] blockdev: sending ioctl 125d to a partition!
[705.460252] blockdev: sending ioctl 125d to a partition!
[705.464216] blockdev: sending ioctl 125d to a partition!
[705.467489] blockdev: sending ioctl 125d to a partition!
[705.468423] blockdev: sending ioctl 125d to a partition!
[705.551779] blockdev: sending ioctl 125d to a partition!
[705.552679] blockdev: sending ioctl 125d to a partition!
done
root@ven:~# _



Finally, I modify my /etc/network/interfaces file, creating a bridge which required the bridge-utils package, and will help us when we get Xen ready later on.



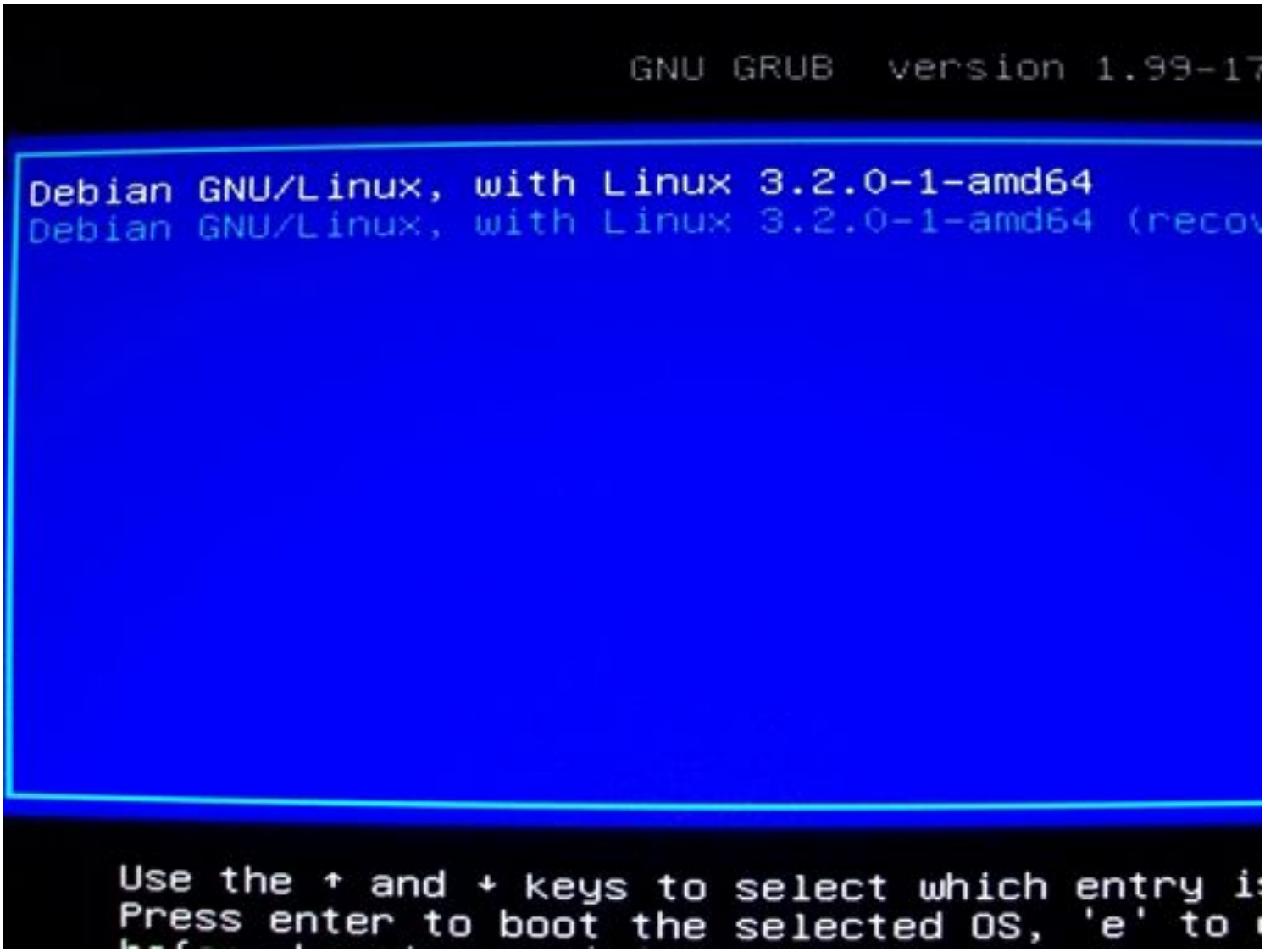

```
# The loopback network interface
auto lo xenbr0
iface lo inet loopback

# The primary network interface
iface eth0 inet manual
iface xenbr0 inet static
        bridge_ports eth0
        address 10.0.1.20
        netmask 255.255.255.0
        network 10.0.1.0
        broadcast 10.0.1.255
        gateway 10.0.1.1
```

I then confirm that the changes worked by restarted my network and ssh:

```
/etc/init.d/networking restart
/etc/init.d/ssh restart
```

Now we can reboot! If you want to confirm the change to your UEFI bios, go ahead and select the boot options and you'll see "debian" among the list now. If all goes well you should get the grub menu:



Give yourself a pat on the back, you just installed Linux without a bootloader, then cleanly introduced a pure EFI solution.

Compiling a Xen Custom Linux Kernel

As a first step be sure you have the following packages installed:

```
sudo aptitude install grub-efi-amd64 sudo ssh bridge-utils parted ntfsprogs bzip2
build-essential libncurses-dev kernel-package fakeroot python-dev uuid-dev libglib2.0-dev
libyajl-dev bcc gcc-multilib iasl libpci-dev mercurial
```

Next run a "df" command to make sure you have at least 6 gigabytes of space available in the directory you plan on compiling.

To speed up the compiling process, my source says you can modify the "/etc/kernel-pkg.conf" file and add a concurrency setting to twice the number of physical cores:

```
CONCURRENCY_LEVEL=8
```

Now we can download our kernel, note that for this guide we are using Kernel 3.3, 3.3.1 from my experience is buggy.

```
wget http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.3.tar.bz2
```

The download will take a few minutes, afterwards we need to extract it:

```
tar jxvf linux-3.3.tar.bz2
```

Enter the folder, and we are ready to configure the kernel:

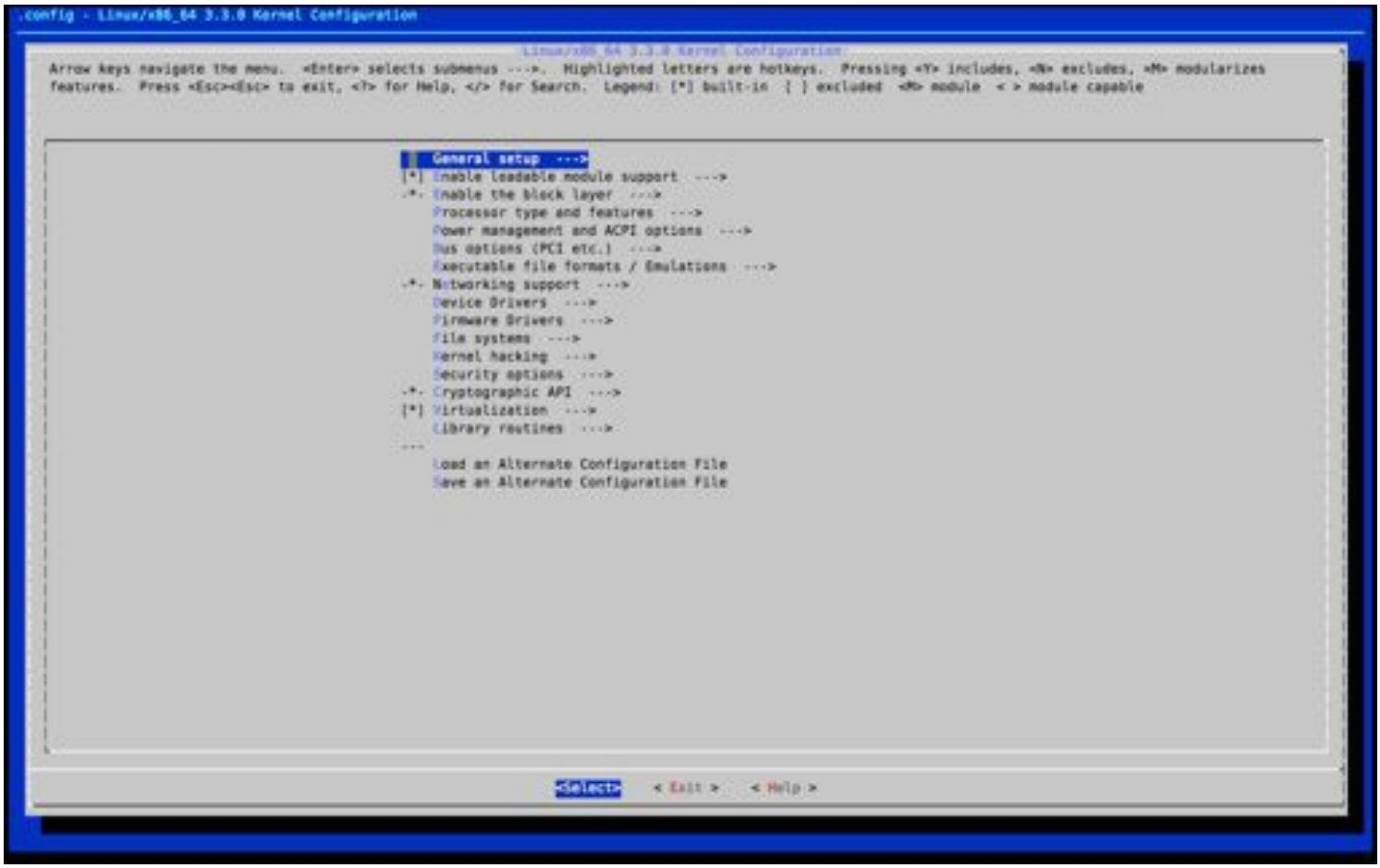
```
cd linux-3.3/
make menuconfig
```

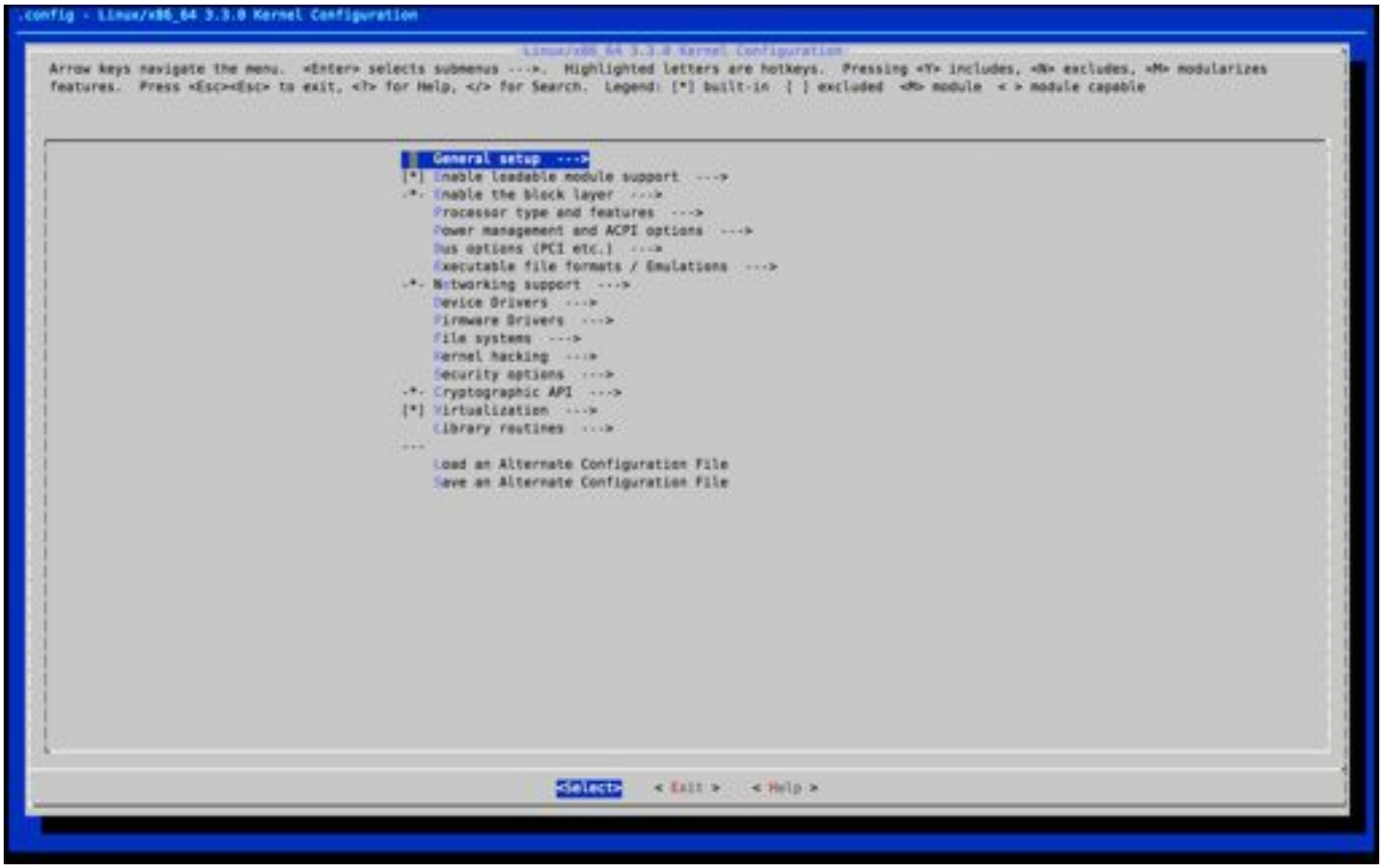
Our objective is specifically to compile a Xen compatible kernel with both EFI and pciback for VGA Passthrough. Here is a hierarchical list of flags and their locations, followed by screenshots:

- Enable the Block Layer
  - Partition Types
    - EFI GUI Partition Support
- Processor Type and Features
  - Paravirtualized Guest Support
    - Xen Guest Support
  - EFI Runtime Service support
  - EFI Stub Support
- Bus Options (PCI etc)
  - Xen PCI Frontend
- Device Drivers
  - Block Devices
    - Xen block-device backend driver
    - Xen virtual block device support
  - Watchdog Timer Support
    - Xen Watchdog Support
  - Xen Driver Support
    - Xen memory balloon driver
    - Memory hotplug support for Xen balloon driver
    - Scrub pages before returning them to system
    - Xen /dev/xen/etchn device
    - Backend driver support
    - Xen Filesystem
    - Create compatibility mount point /proc/xen
    - Create xen entries under /sys/hypervisor
    - userspace grant access device driver
    - User-space grant reference allocator driver
    - Xen PCI-device backend driver
  - IOMMU Hardware Support

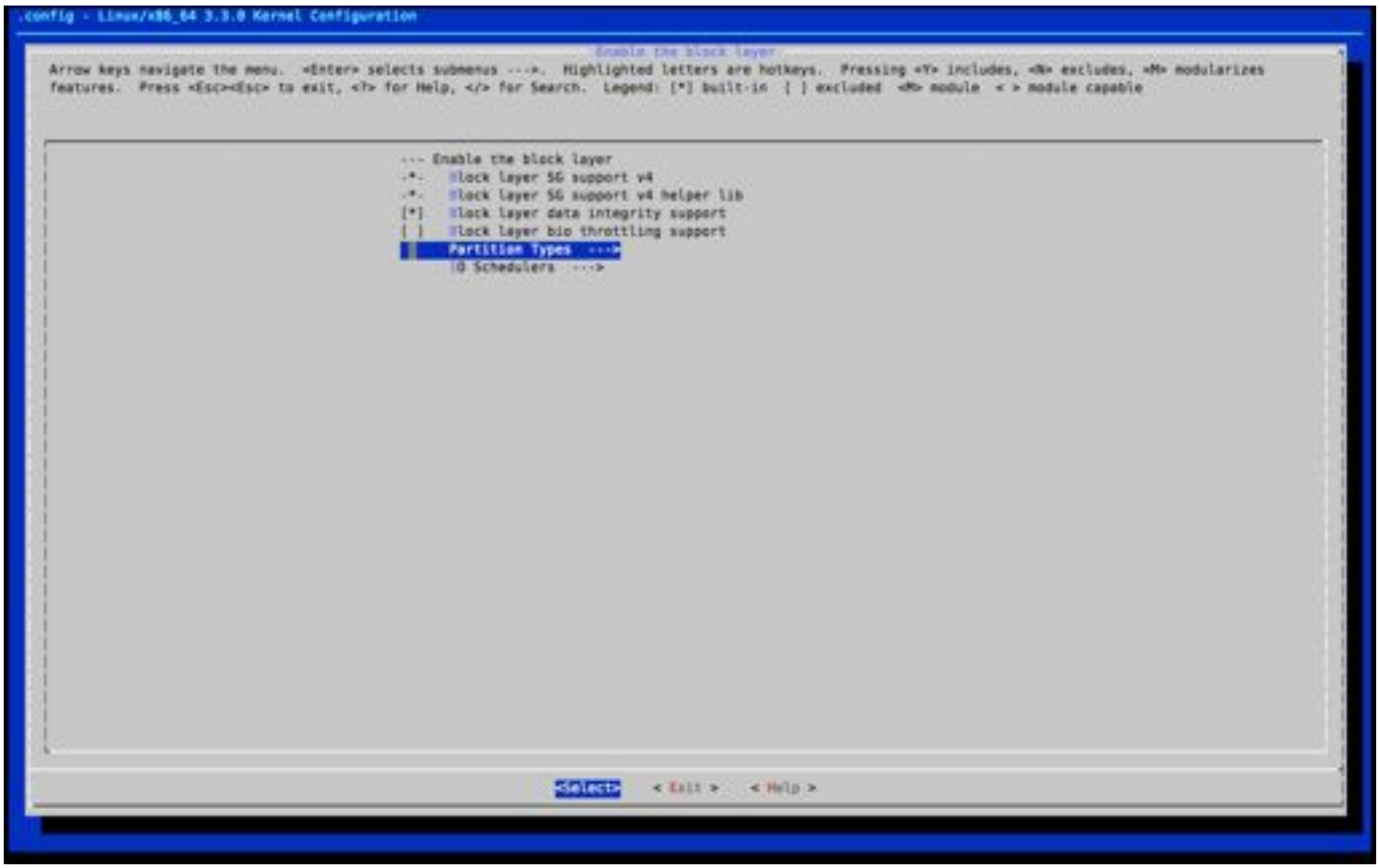
        - Enable Intel DMA Remapping Devices by default

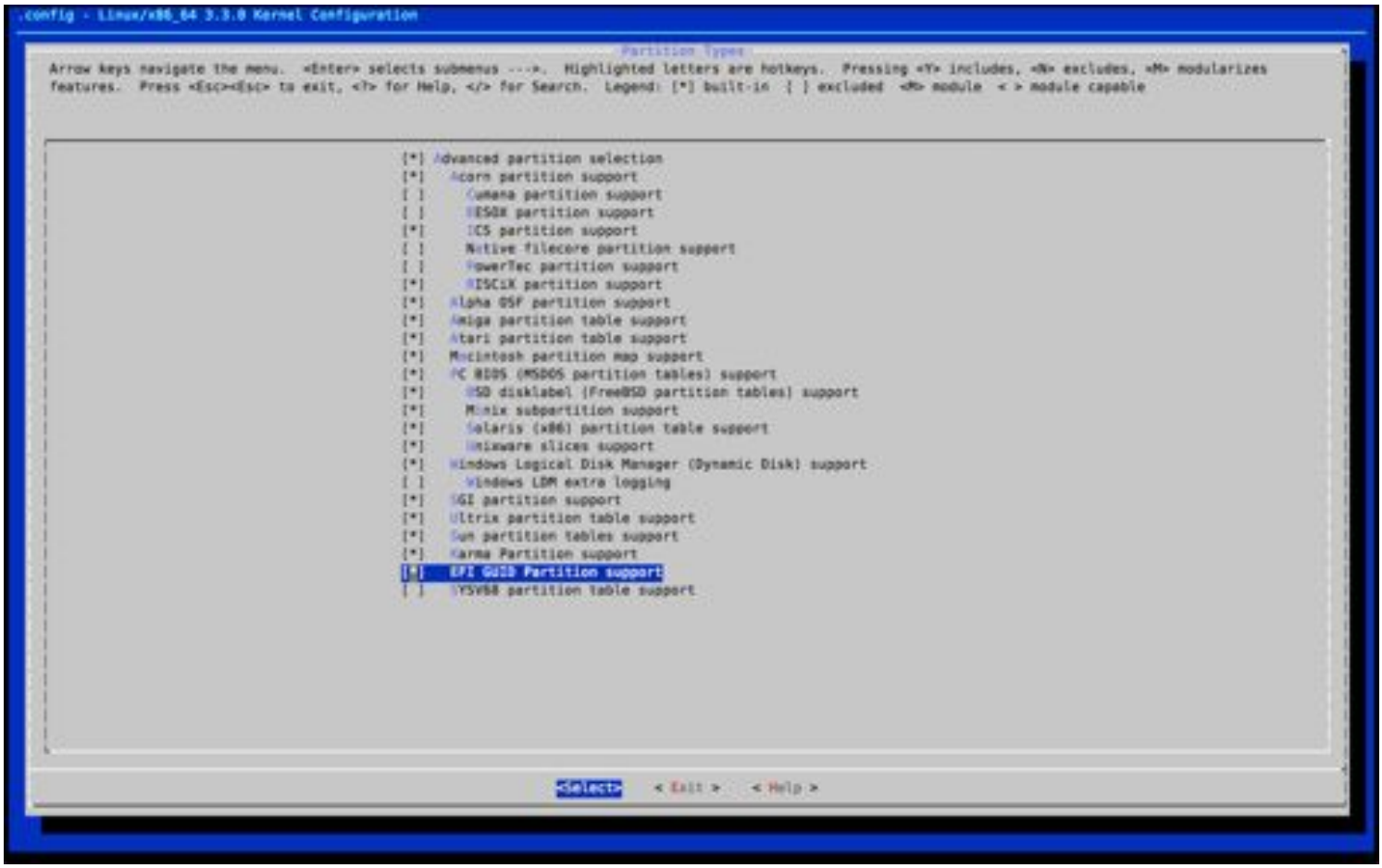
- Firmware Drivers
    - EFI Variable Support via sysfs

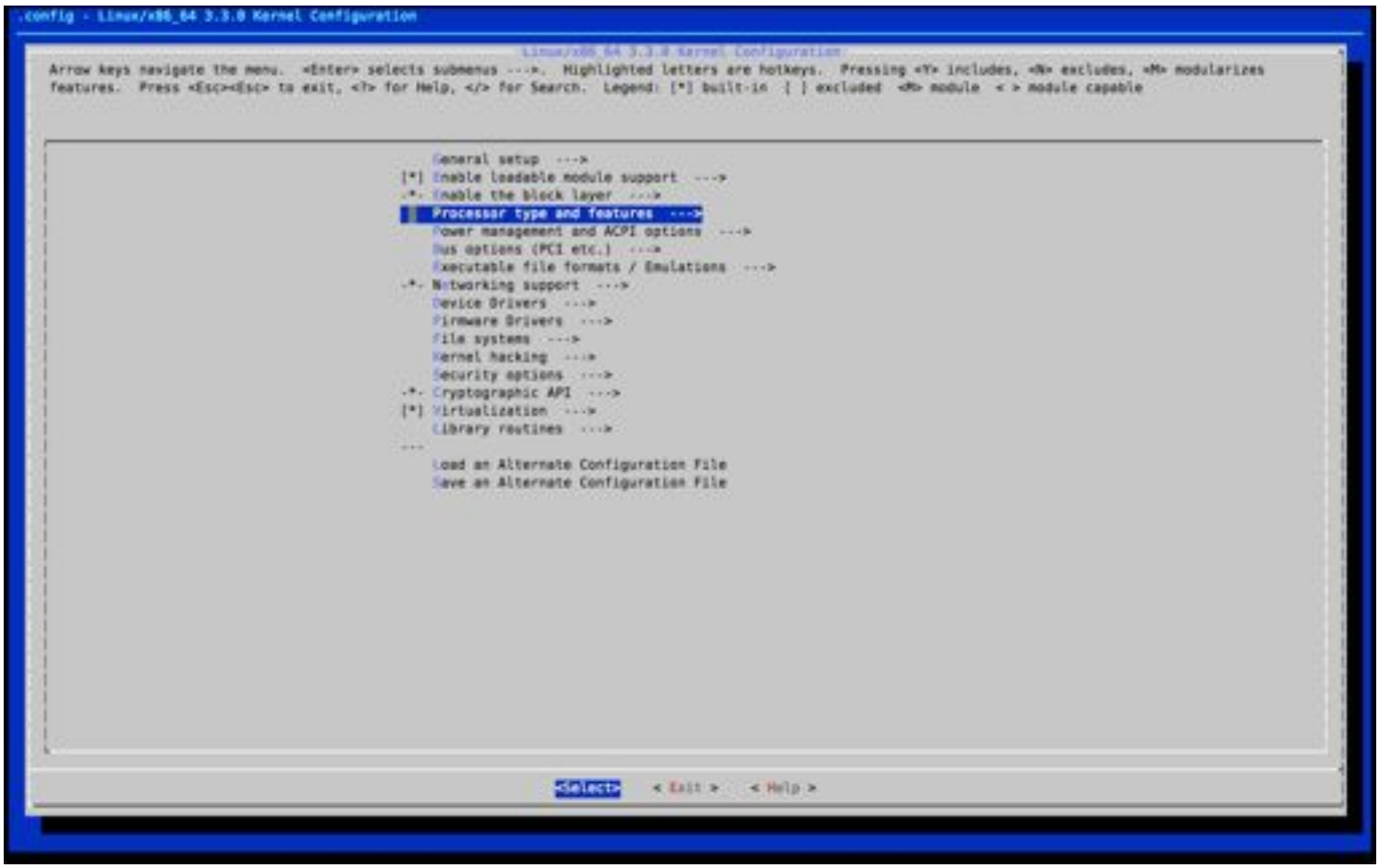



.config - Linux/x86_64 3.3.0 Kernel Configuration
Linux/x86_64 3.3.0 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
General setup --->
[*] Enable loadable module support --->
-*- Enable the block layer --->
Processor type and features --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Executable file formats / Emulations --->
-*- Networking support --->
Device Drivers --->
Firmware Drivers --->
File systems --->
Kernel hacking --->
Security options --->
-*- Cryptographic API --->
[*] Virtualization --->
Library routines --->

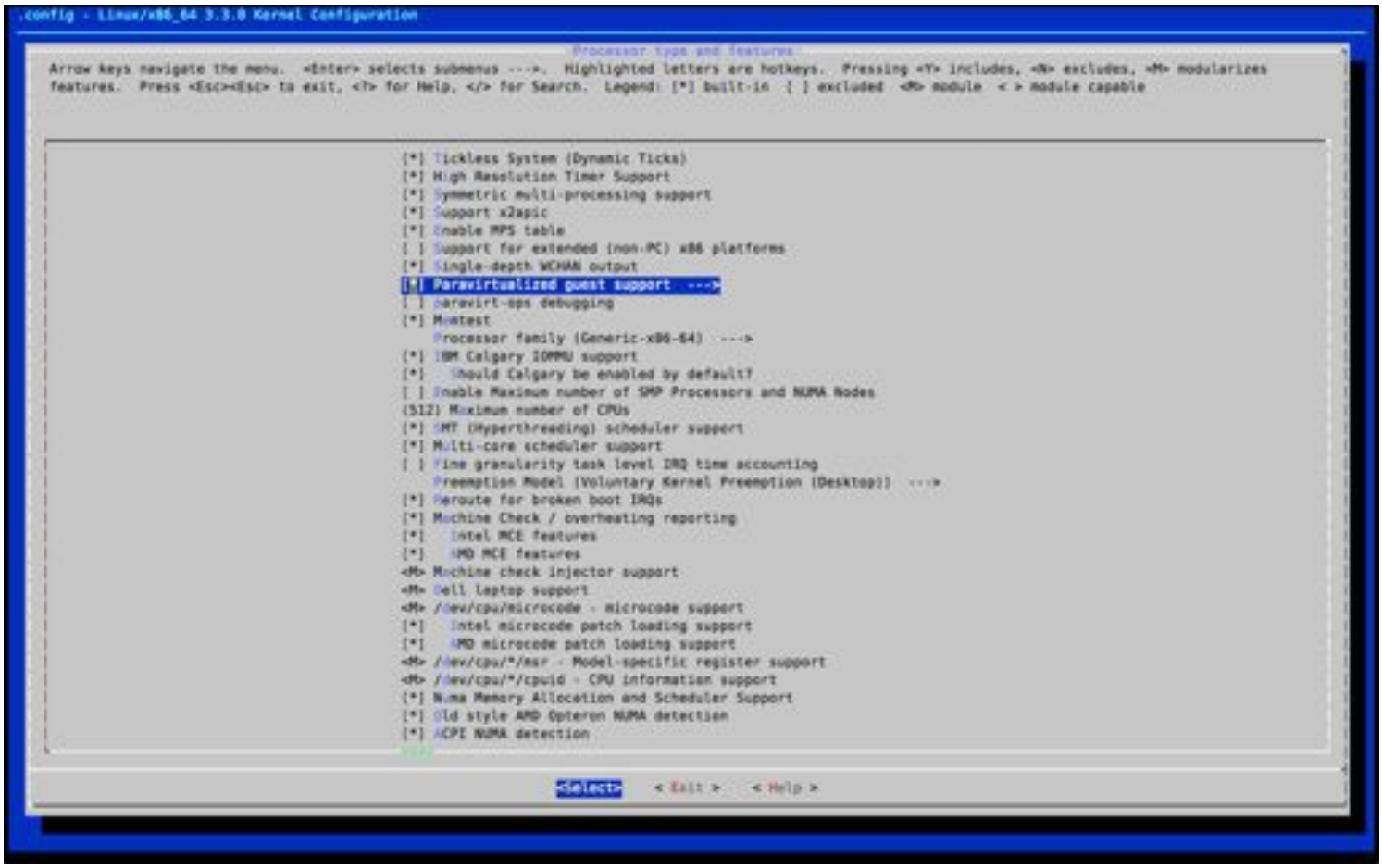
Load an Alternate Configuration File
Save an Alternate Configuration File
<Select> < Exit > < Help >

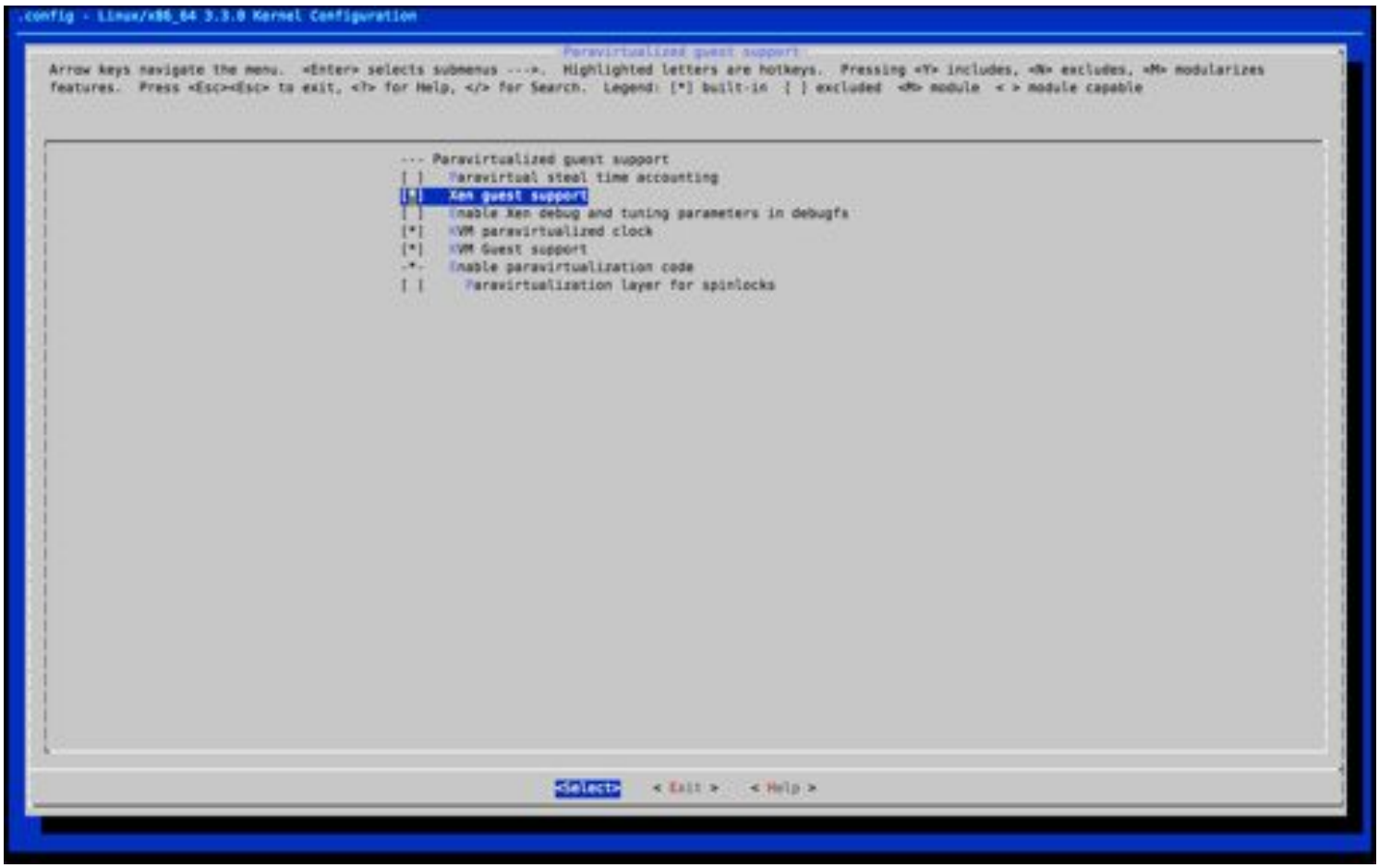

.config - Linux/x86_64 3.3.0 Kernel Configuration
Enable the block layer
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
--- Enable the block layer
-*- Block layer SG support v4
-*- Block layer SG support v4 helper lib
[*] Block layer data integrity support
[ ] Block layer bio throttling support
Partition Types --->
IO Schedulers --->
<Select> < Exit > < Help >

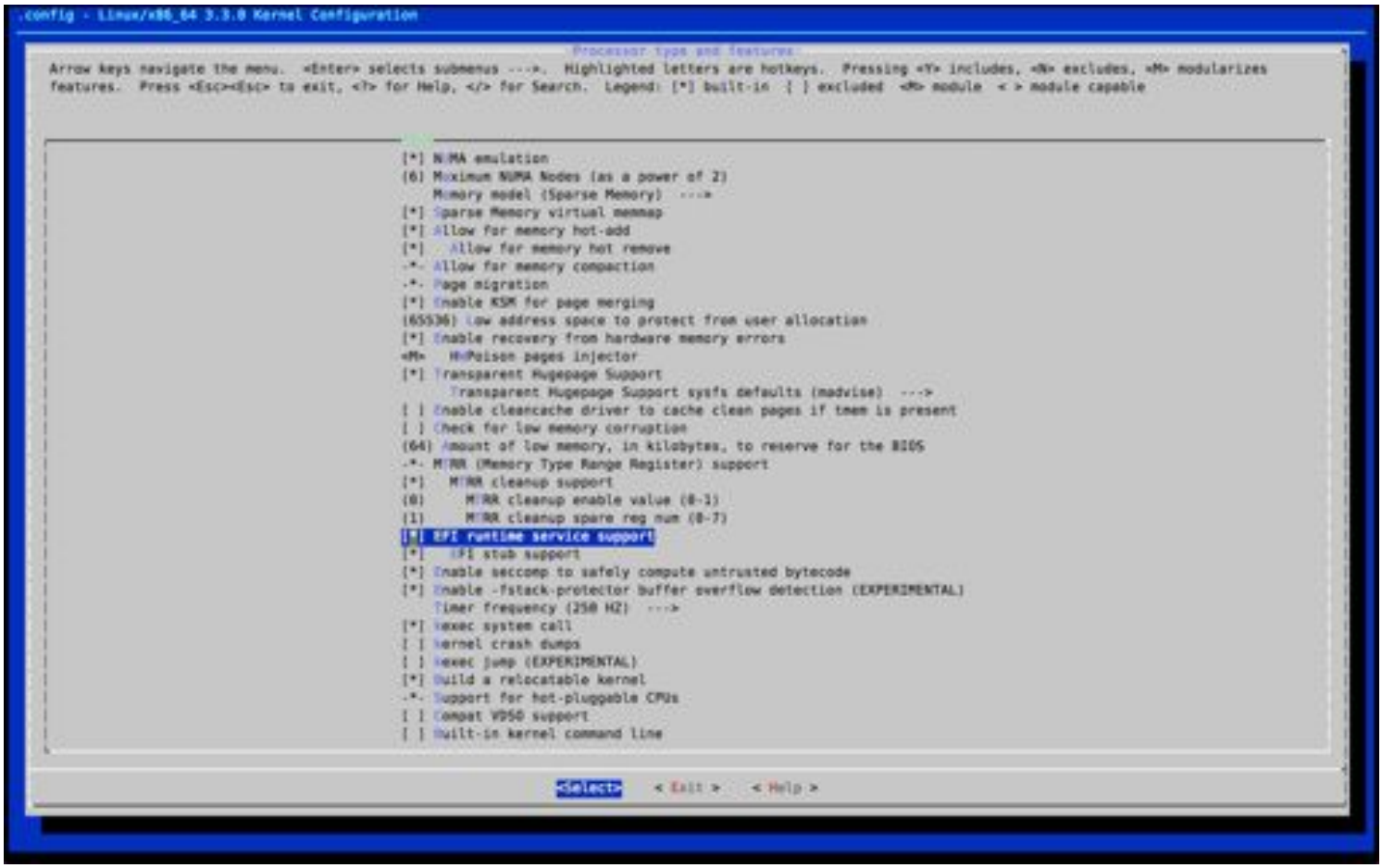

.config - Linux/x86_64 3.3.0 Kernel Configuration
Partition Types
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] Advanced partition selection
[*] Acorn partition support
[ ] Cumana partition support
[ ] EESOX partition support
[*] ICS partition support
[ ] Native filecore partition support
[ ] PowerTec partition support
[*] RISCiX partition support
[*] Alpha OSF partition support
[*] Amiga partition table support
[*] Atari partition table support
[*] Macintosh partition map support
[*] PC BIOS (MSDOS partition tables) support
[*] BSD disklabel (FreeBSD partition tables) support
[*] Minix subpartition support
[*] Solaris (x86) partition table support
[*] Unixware slices support
[*] Windows Logical Disk Manager (Dynamic Disk) support
[ ] Windows LDM extra logging
[*] SGI partition support
[*] Ultrix partition table support
[*] Sun partition tables support
[*] Karma Partition support
[*] EFI GUID Partition support
[ ] SYSV68 partition table support
<Select> < Exit > < Help >

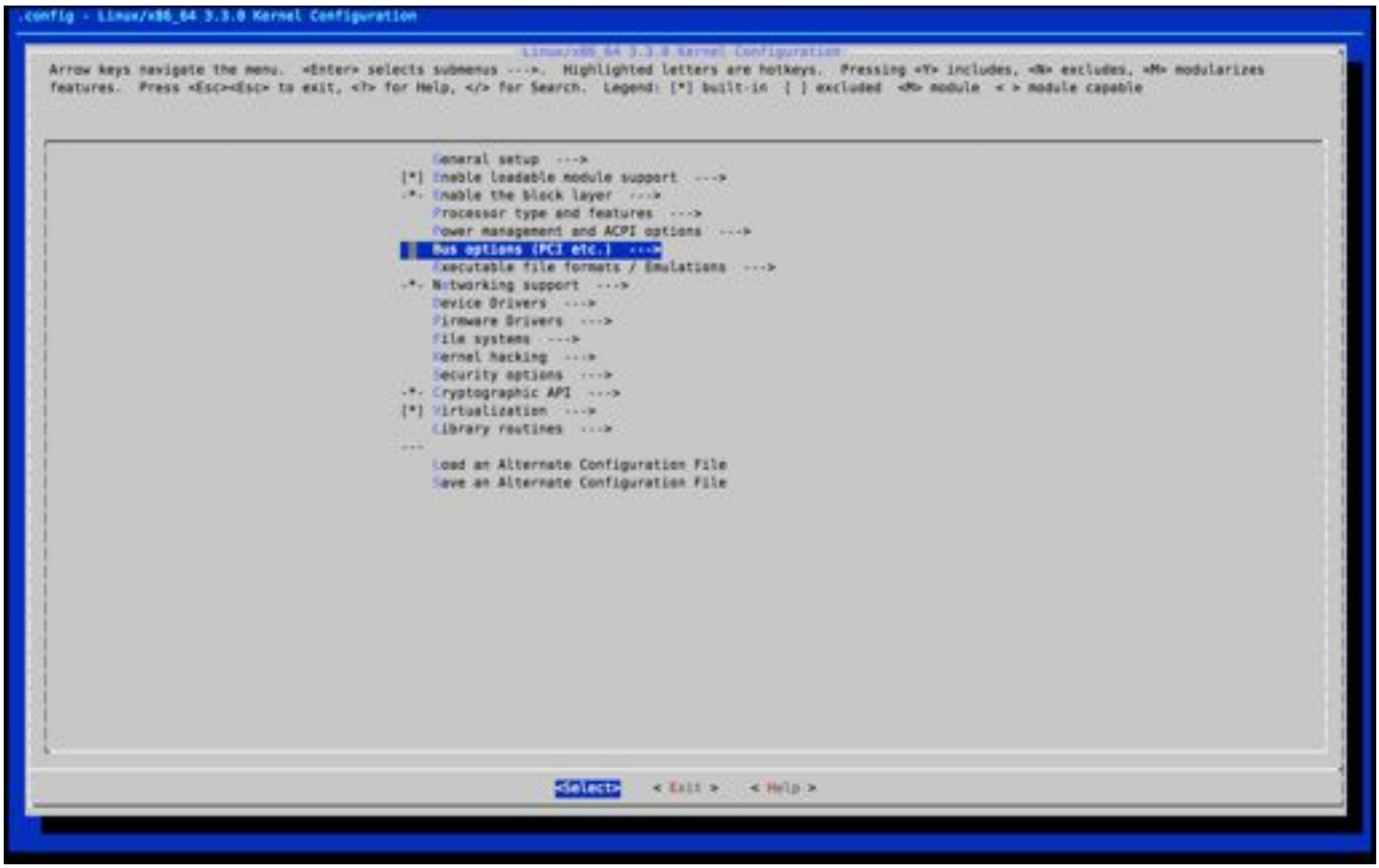

.config - Linux/x86_64 3.3.0 Kernel Configuration
Linux/x86_64 3.3.0 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
General setup --->
[*] Enable loadable module support --->
-*- Enable the block layer --->
Processor type and features --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Executable file formats / Emulations --->
-*- Networking support --->
Device Drivers --->
Firmware Drivers --->
File systems --->
Kernel hacking --->
Security options --->
-*- Cryptographic API --->
[*] Virtualization --->
Library routines --->

Load an Alternate Configuration File
Save an Alternate Configuration File
<Select> < Exit > < Help >

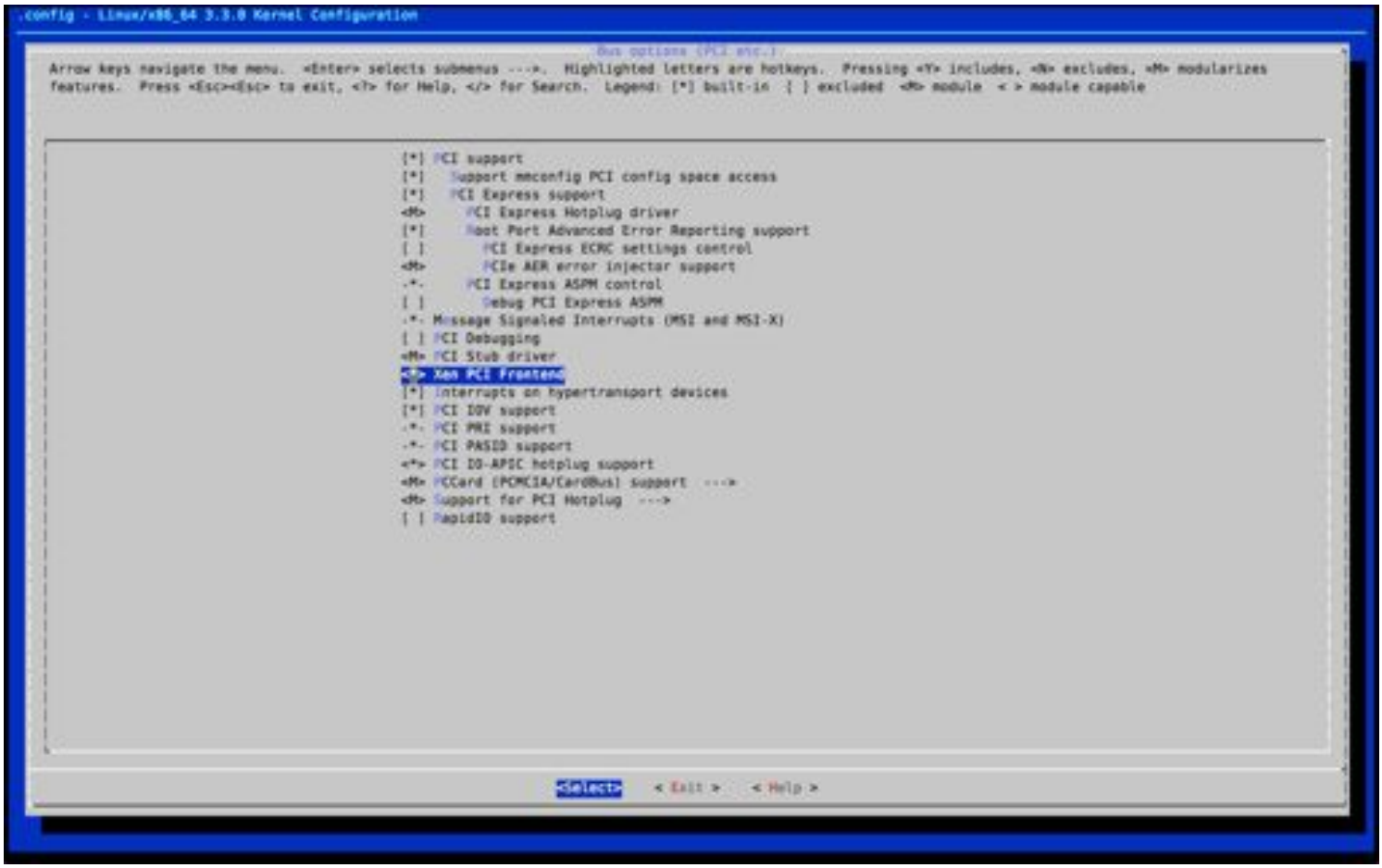

.config - Linux/x86_64 3.3.0 Kernel Configuration
Processor type and features
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] Tickless System (Dynamic Ticks)
[*] High Resolution Timer Support
[*] Symmetric multi-processing support
[*] Support x2apic
[*] Enable MPS table
[ ] Support for extended (non-PC) x86 platforms
[*] Single-depth WCHAN output
[*] Paravirtualized guest support --->
[ ] Paravirt-ops debugging
[*] Memtest
Processor family (Generic-x86-64) --->
[*] IBM Calgary IOMMU support
[*] Should Calgary be enabled by default?
[ ] Enable Maximum number of SMP Processors and NUMA Nodes
(512) Maximum number of CPUs
[*] SMT (Hyperthreading) scheduler support
[*] Multi-core scheduler support
[ ] Fine granularity task level IRQ time accounting
Preemption Model (Voluntary Kernel Preemption (Desktop)) --->
[*] Reroute for broken boot IRQs
[*] Machine Check / overheating reporting
[*] Intel MCE features
[*] AMD MCE features
<M> Machine check injector support
<M> Dell laptop support
<M> /dev/cpu/microcode - microcode support
[*] Intel microcode patch loading support
[*] AMD microcode patch loading support
<M> /dev/cpu/*/msr - Model-specific register support
<M> /dev/cpu/*/cpuid - CPU information support
[*] Numa Memory Allocation and Scheduler Support
[*] Old style AMD Opteron NUMA detection
[*] ACPI NUMA detection
<Select> < Exit > < Help >

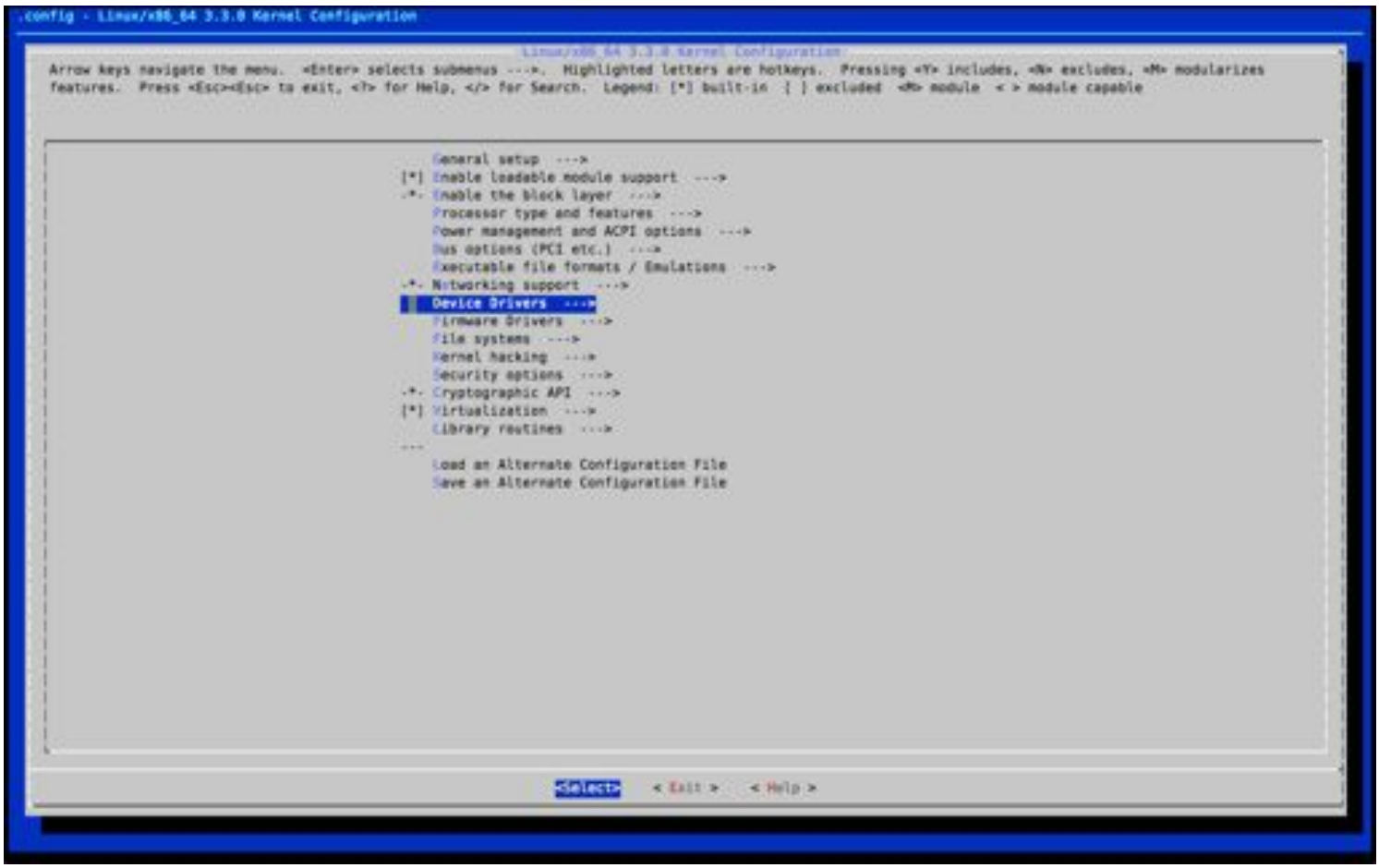

.config - Linux/x86_64 3.3.0 Kernel Configuration
Paravirtualized guest support
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
--- Paravirtualized guest support
[ ] Paravirtual steal time accounting
[*] Xen guest support
[ ] Enable Xen debug and tuning parameters in debugfs
[*] KVM paravirtualized clock
[*] KVM Guest support
-*- Enable paravirtualization code
[ ] Paravirtualization layer for spinlocks
<Select> < Exit > < Help >


.config - Linux/x86_64 3.3.0 Kernel Configuration
Processor type and features
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] NUMA emulation
(6) Maximum NUMA Nodes (as a power of 2)
Memory model (Sparse Memory) --->
[*] Sparse Memory virtual memmap
[*] Allow for memory hot-add
[*] Allow for memory hot remove
-*- Allow for memory compaction
-*- Page migration
[*] Enable KSM for page merging
(65536) Low address space to protect from user allocation
[*] Enable recovery from hardware memory errors
<M> HWPoison pages injector
[*] Transparent Hugepage Support
Transparent Hugepage Support sysfs defaults (madvise) --->
[ ] Enable cleancache driver to cache clean pages if tmem is present
[ ] Check for low memory corruption
(64) Amount of low memory, in kilobytes, to reserve for the BIOS
-*- MTRR (Memory Type Range Register) support
[*] MTRR cleanup support
(0) MTRR cleanup enable value (0-1)
(1) MTRR cleanup spare reg num (0-7)
[*] EFI runtime service support
[*] EFI stub support
[*] Enable seccomp to safely compute untrusted bytecode
[*] Enable -fstack-protector buffer overflow detection (EXPERIMENTAL)
Timer frequency (250 HZ) --->
[*] kexec system call
[ ] kernel crash dumps
[ ] kexec jump (EXPERIMENTAL)
[*] Build a relocatable kernel
-*- Support for hot-pluggable CPUs
[ ] Compat VDSO support
[ ] Built-in kernel command line
<Select> < Exit > < Help >

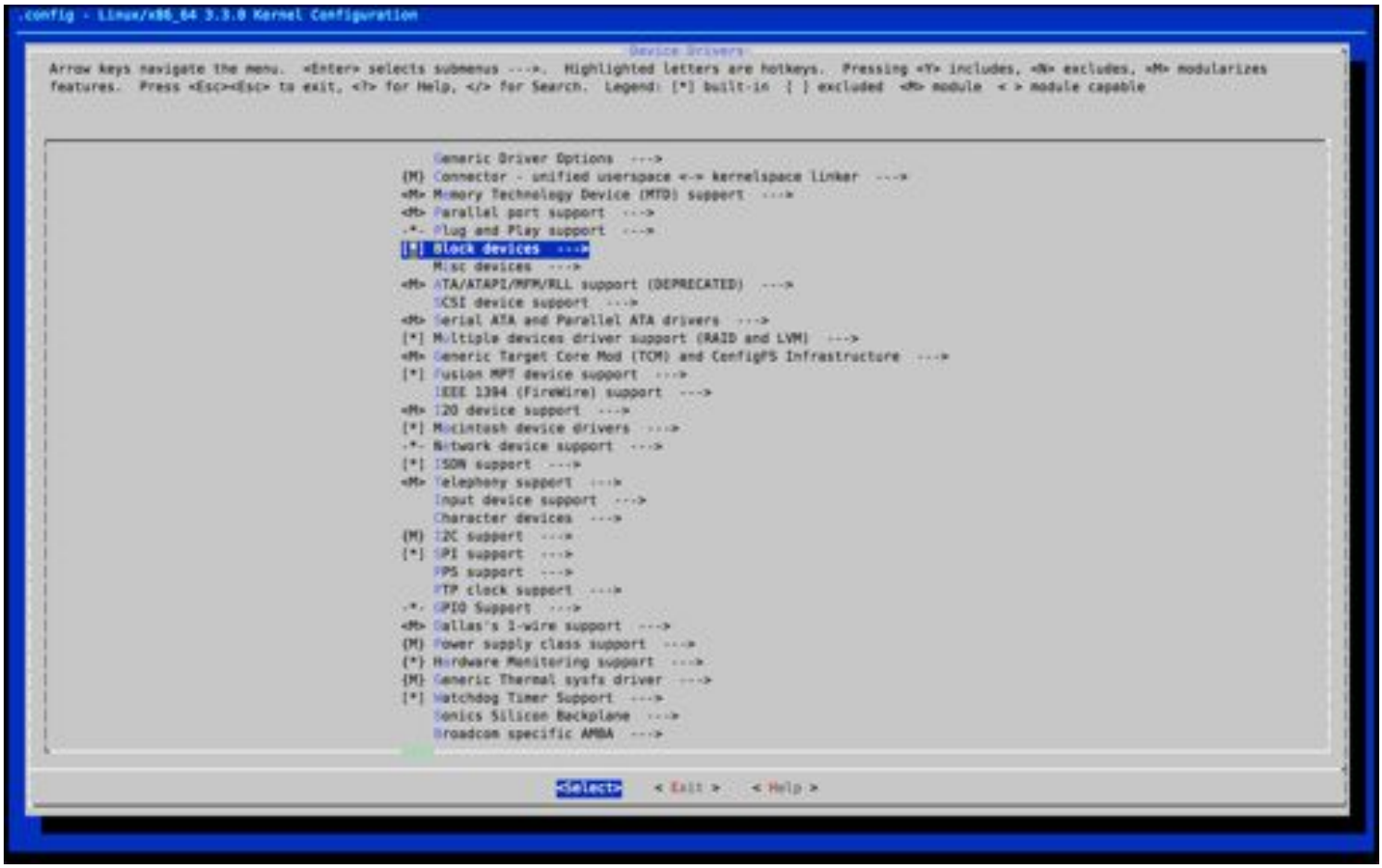

.config - Linux/x86_64 3.3.0 Kernel Configuration
Linux/x86_64 3.3.0 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
General setup --->
[*] Enable loadable module support --->
-*- Enable the block layer --->
Processor type and features --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Executable file formats / Emulations --->
-*- Networking support --->
Device Drivers --->
Firmware Drivers --->
File systems --->
Kernel hacking --->
Security options --->
-*- Cryptographic API --->
[*] Virtualization --->
Library routines --->

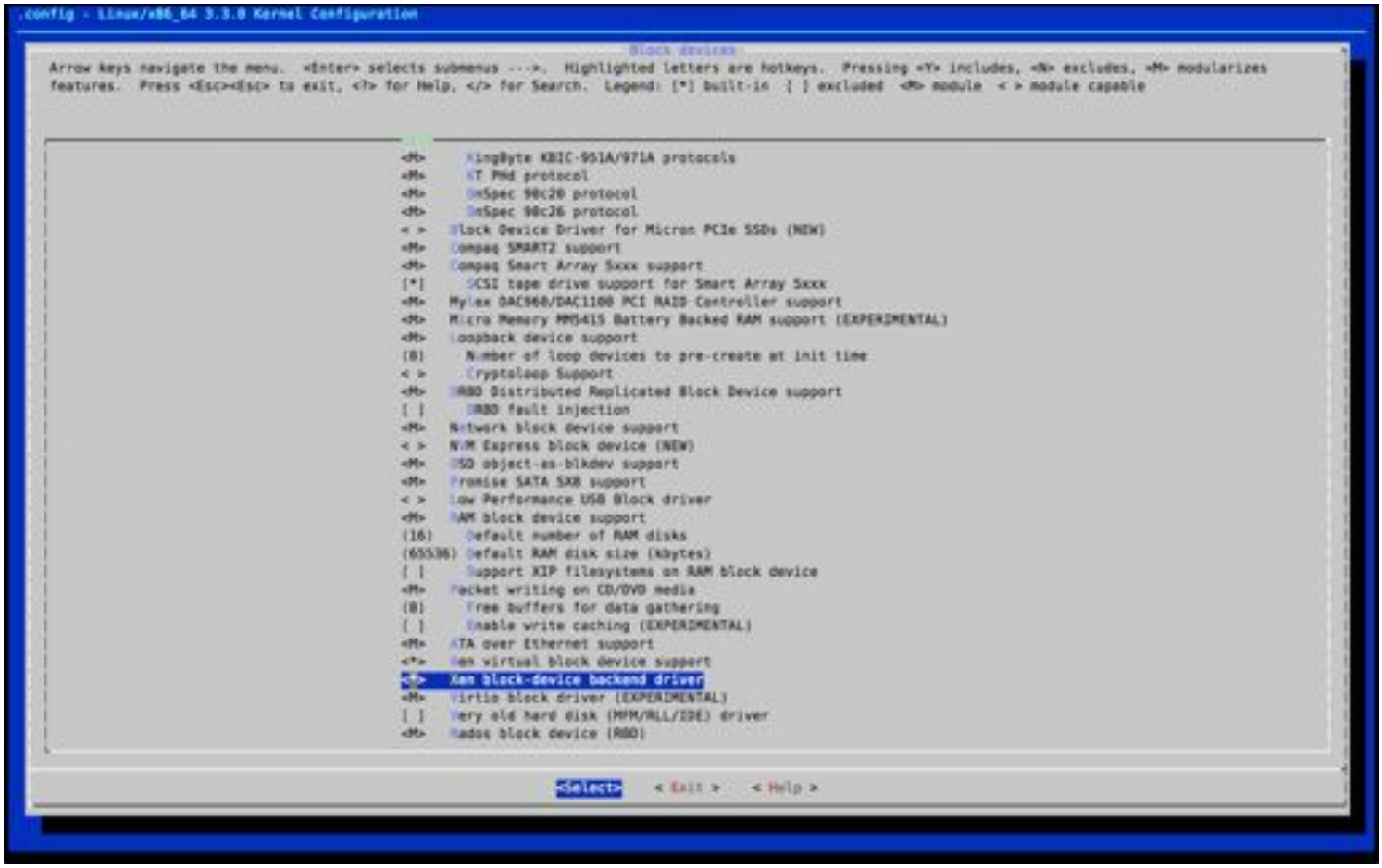
Load an Alternate Configuration File
Save an Alternate Configuration File
<Select> < Exit > < Help >

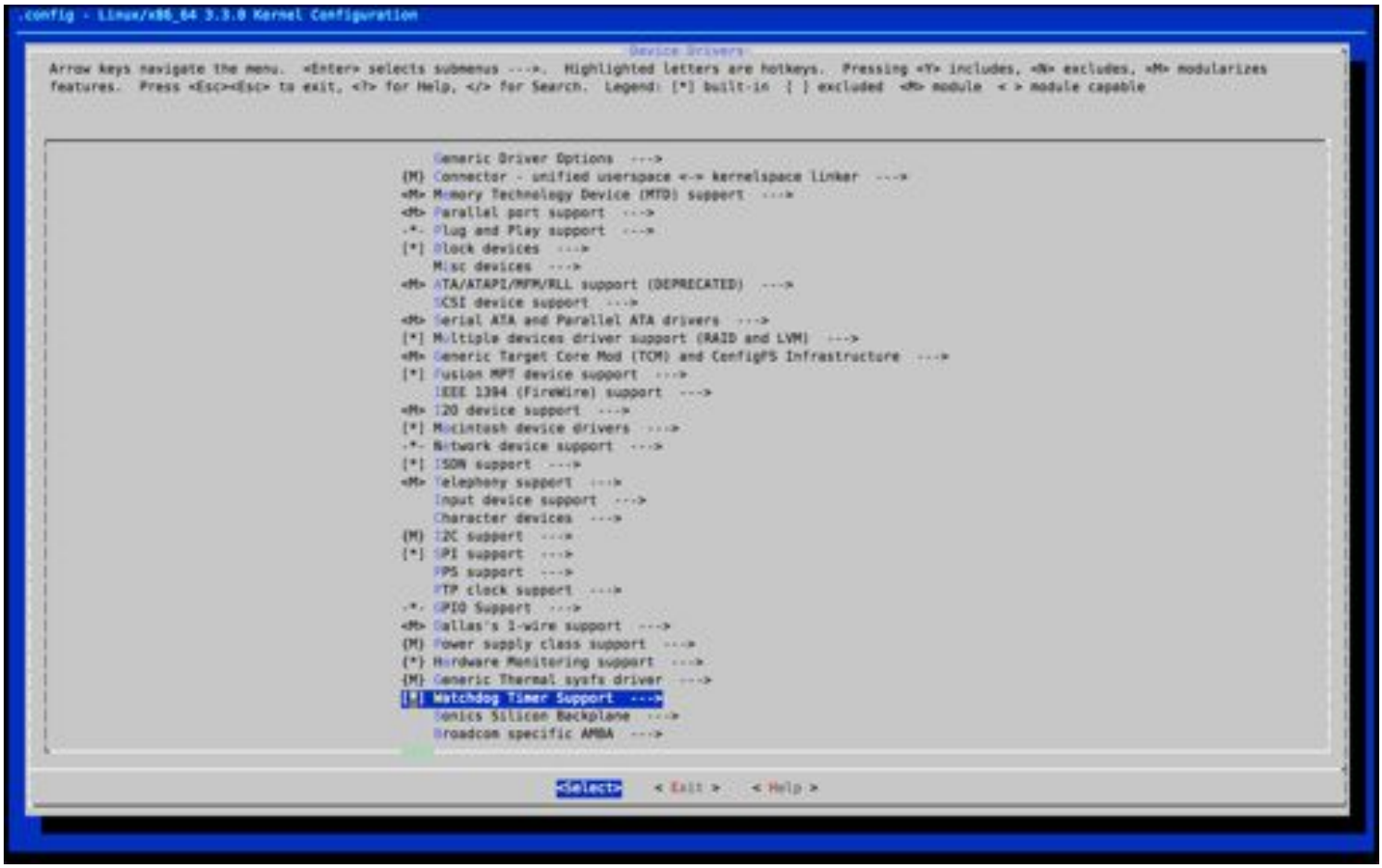

.config - Linux/x86_64 3.3.0 Kernel Configuration
Bus options (PCI etc.)
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] PCI support
[*] Support mmconfig PCI config space access
[*] PCI Express support
<M> PCI Express Hotplug driver
[*] Root Port Advanced Error Reporting support
[ ] PCI Express ECRC settings control
<M> PCIe AER error injectar support
-*- PCI Express ASPM control
[ ] Debug PCI Express ASPM
-*- Message Signaled Interrupts (MSI and MSI-X)
[ ] PCI Debugging
<M> PCI Stub driver
<*> Xen PCI Frontend
[*] Interrupts on hypertransport devices
[*] PCI IOV support
-*- PCI PRI support
-*- PCI PASID support
<*> PCI IO-APIC hotplug support
<M> PCCard (PCMCIA/CardBus) support --->
<M> Support for PCI Hotplug --->
[ ] RapidIO support
<Select> < Exit > < Help >

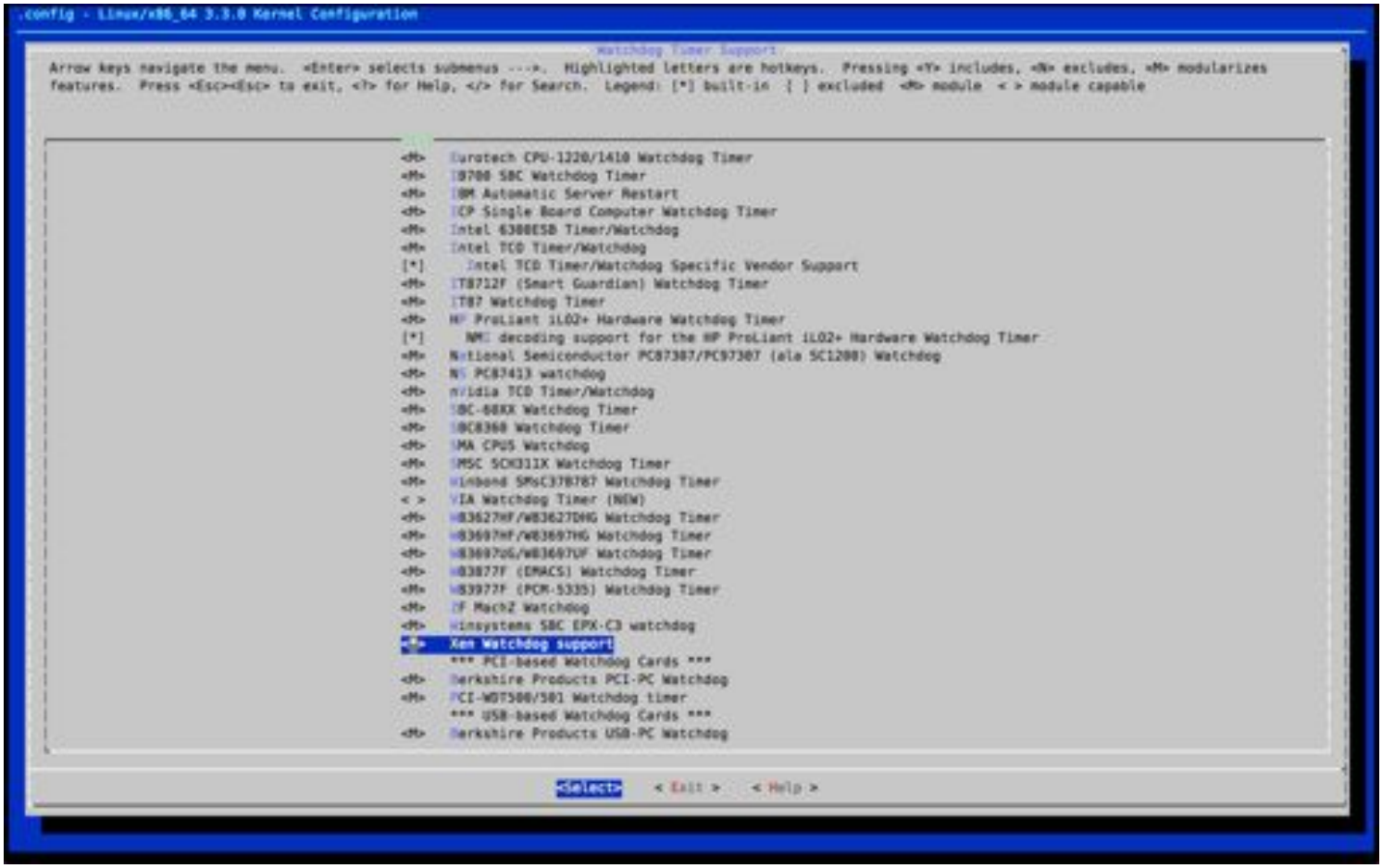

.config - Linux/x86_64 3.3.0 Kernel Configuration
Linux/x86_64 3.3.0 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
General setup --->
[*] Enable loadable module support --->
-*- Enable the block layer --->
Processor type and features --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Executable file formats / Emulations --->
-*- Networking support --->
Device Drivers --->
Firmware Drivers --->
File systems --->
Kernel hacking --->
Security options --->
-*- Cryptographic API --->
[*] Virtualization --->
Library routines --->

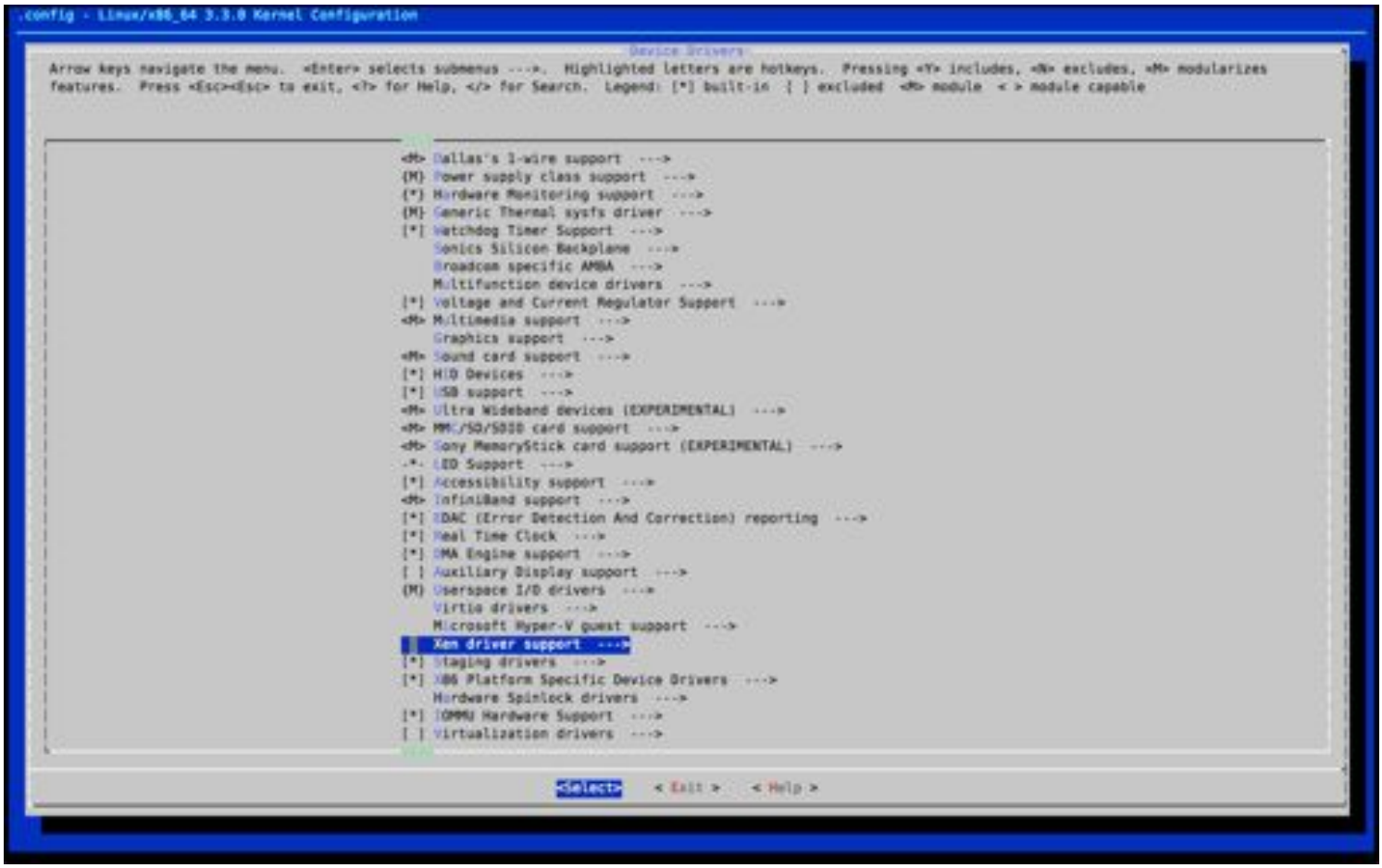
Load an Alternate Configuration File
Save an Alternate Configuration File
<Select> < Exit > < Help >

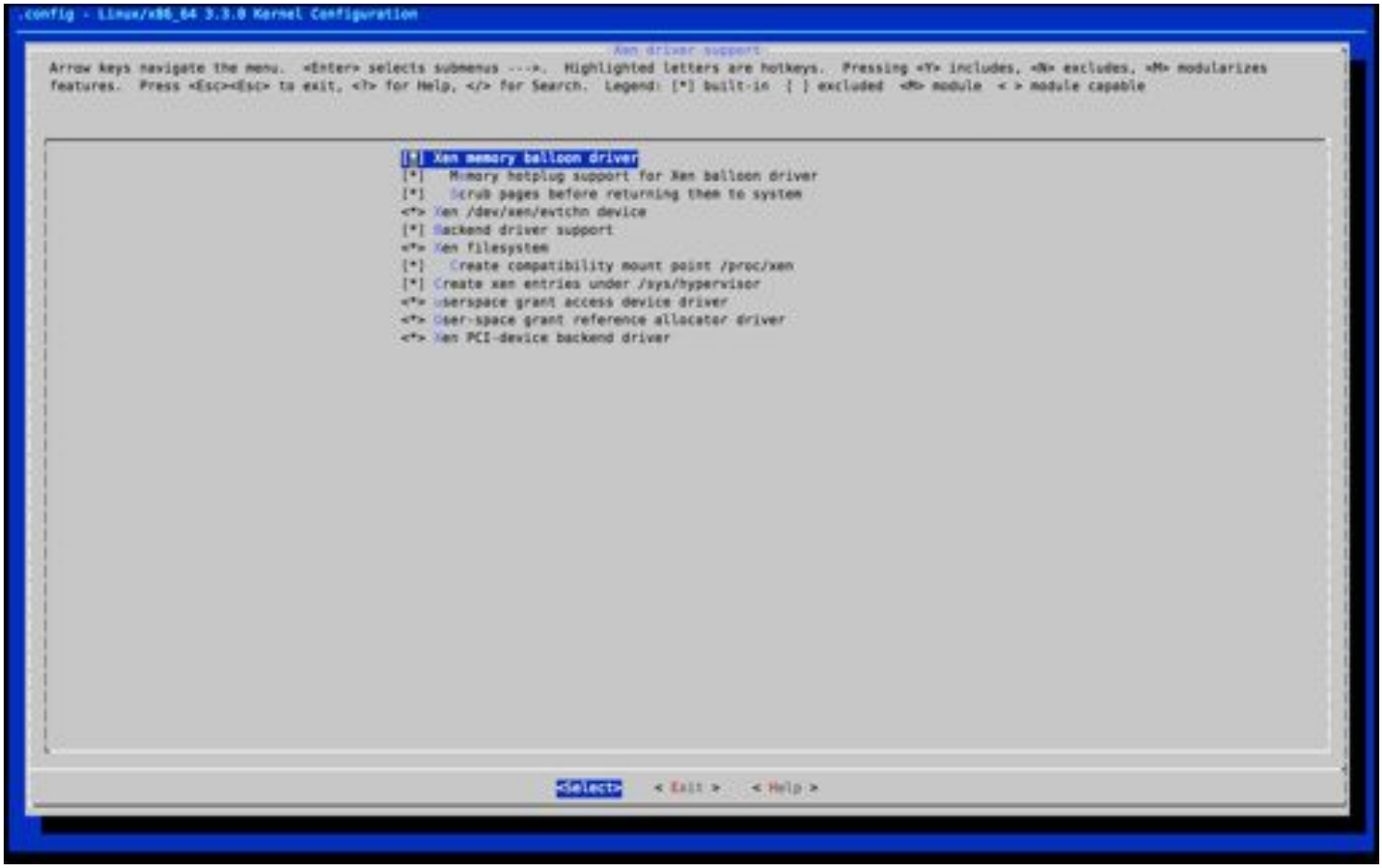

.config - Linux/x86_64 3.3.0 Kernel Configuration
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
Generic Driver Options --->
{M} Connector - unified userspace <-> kernelspace linker --->
<M> Memory Technology Device (MTD) support --->
<M> Parallel port support --->
-*- Plug and Play support --->
[*] Block devices --->
Misc devices --->
<M> ATA/ATAPI/MFM/RLL support (DEPRECATED) --->
SCSI device support --->
<M> Serial ATA and Parallel ATA drivers --->
[*] Multiple devices driver support (RAID and LVM) --->
<M> Generic Target Core Mod (TCM) and ConfigFS Infrastructure --->
[*] Fusion MPT device support --->
IEEE 1394 (FireWire) support --->
<M> I2O device support --->
[*] Macintosh device drivers --->
-*- Network device support --->
[*] ISDN support --->
<M> Telephony support --->
Input device support --->
Character devices --->
{M} I2C support --->
[*] SPI support --->
PPS support --->
PTP clock support --->
-*- GPIO Support --->
<M> Dallas's 1-wire support --->
{M} Power supply class support --->
{*} Hardware Monitoring support --->
{M} Generic Thermal sysfs driver --->
[*] Watchdog Timer Support --->
Sonics Silicon Backplane --->
Broadcom specific AMBA --->
<Select> < Exit > < Help >

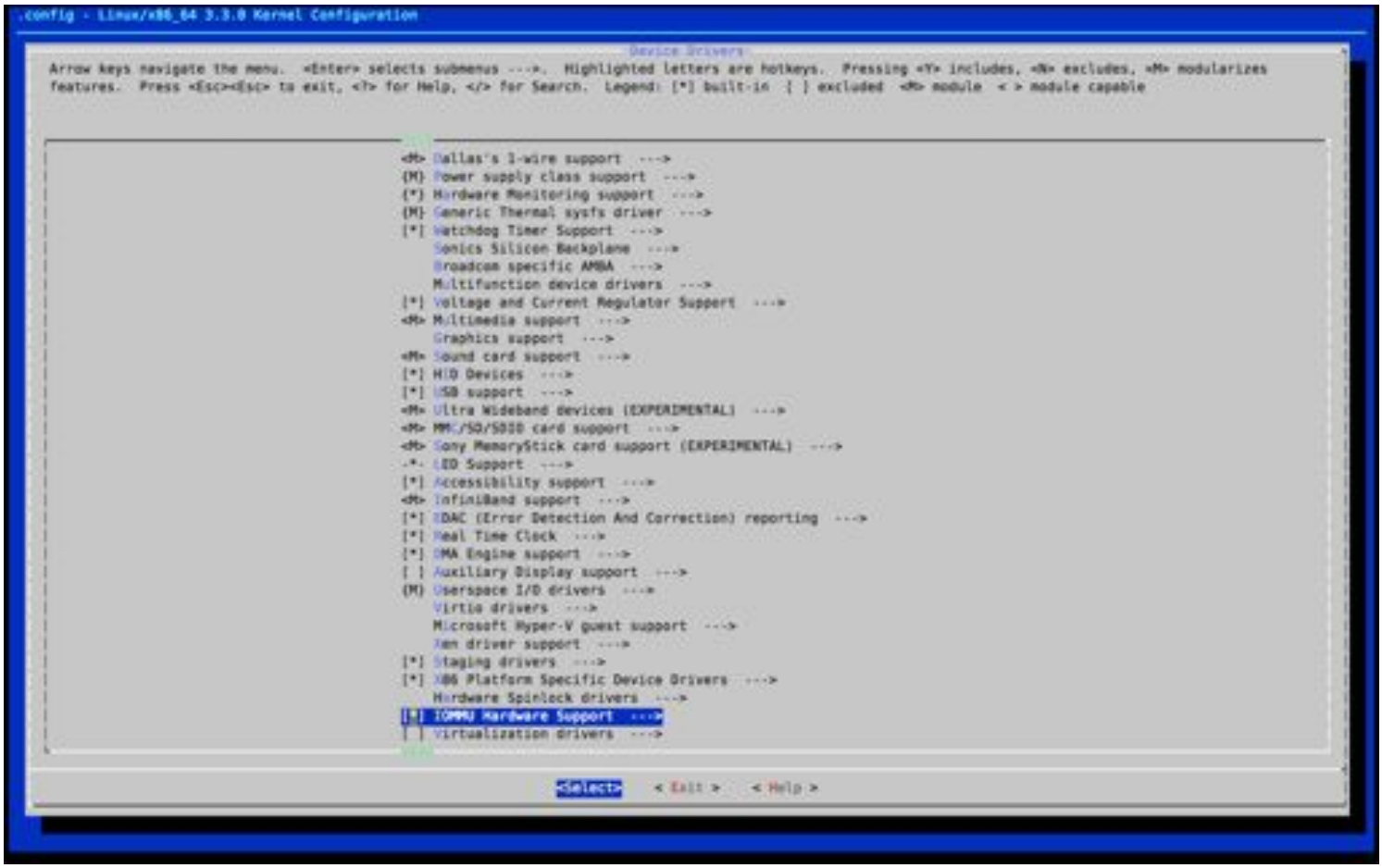

.config - Linux/x86_64 3.3.0 Kernel Configuration
Block devices
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<M> KingByte KBIC-951A/971A protocols
<M> XT PMd protocol
<M> OnSpec 90c20 protocol
<M> OnSpec 90c26 protocol
< > Block Device Driver for Micron PCIe SSDs (NEW)
<M> Compaq SMART2 support
<M> Compaq Smart Array 5xxx support
[*] SCSI tape drive support for Smart Array 5xxx
<M> Mylex DAC960/DAC1100 PCI RAID Controller support
<M> Micro Memory MM5415 Battery Backed RAM support (EXPERIMENTAL)
<M> Loopback device support
(8) Number of loop devices to pre-create at init time
< > Cryptoloop Support
<M> DRBD Distributed Replicated Block Device support
[ ] DRBD fault injection
<M> Network block device support
< > NVM Express block device (NEW)
<M> OSD object-as-blkdev support
<M> Promise SATA SX8 support
< > Low Performance USB Block driver
<M> RAM block device support
(16) Default number of RAM disks
(65536) Default RAM disk size (kbytes)
[ ] Support XIP filesystems on RAM block device
<M> Packet writing on CD/DVD media
(8) Free buffers for data gathering
[ ] Enable write caching (EXPERIMENTAL)
<M> ATA over Ethernet support
<*> Xen virtual block device support
<*> Xen block-device backend driver
<M> Virtio block driver (EXPERIMENTAL)
[ ] Very old hard disk (MFM/RLL/IDE) driver
<M> Rados block device (RBD)
<Select> < Exit > < Help >

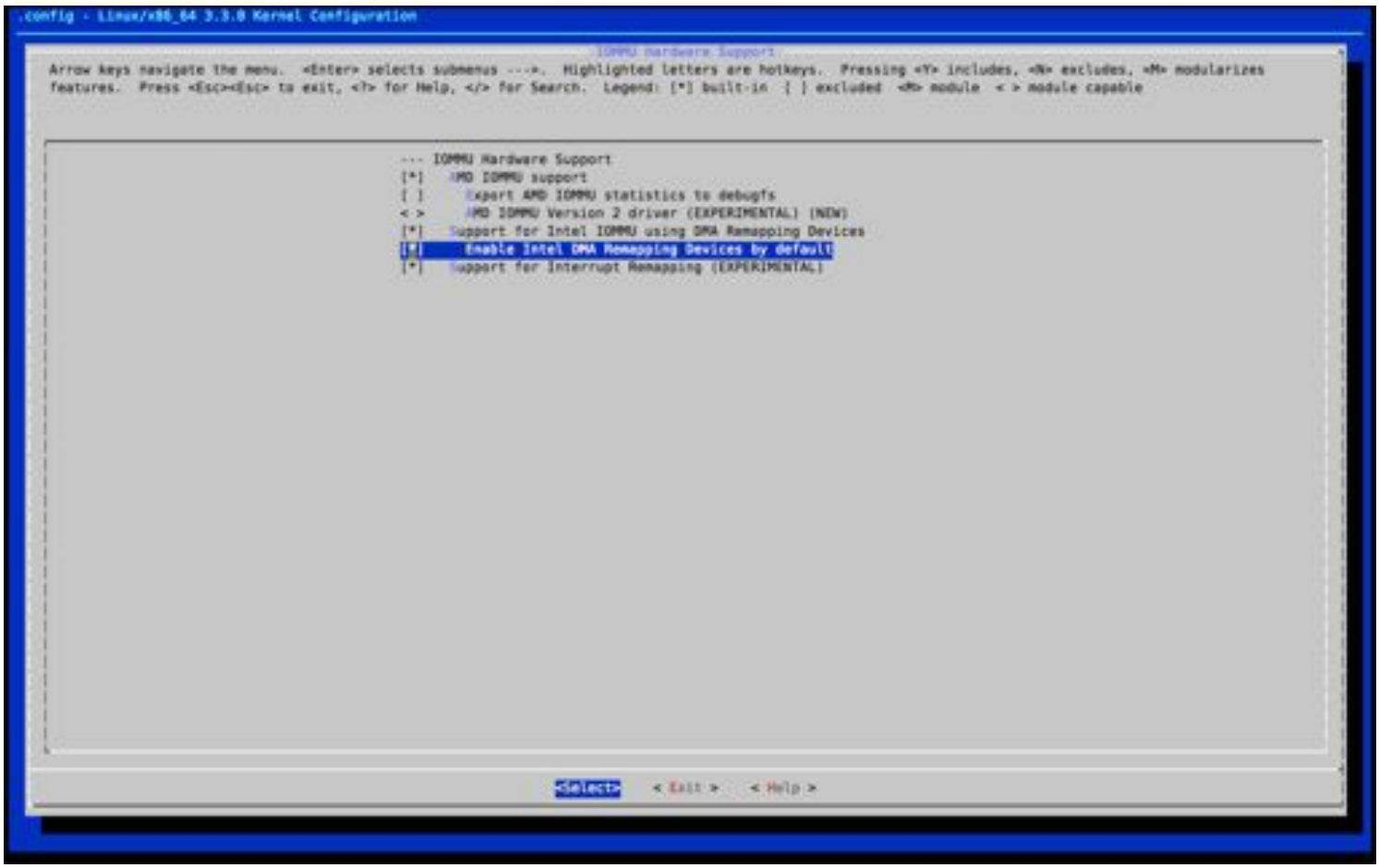

.config - Linux/x86_64 3.3.0 Kernel Configuration
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
Generic Driver Options --->
{M} Connector - unified userspace <-> kernelspace linker --->
<M> Memory Technology Device (MTD) support --->
<M> Parallel port support --->
-*- Plug and Play support --->
[*] Block devices --->
Misc devices --->
<M> ATA/ATAPI/MFM/RLL support (DEPRECATED) --->
SCSI device support --->
<M> Serial ATA and Parallel ATA drivers --->
[*] Multiple devices driver support (RAID and LVM) --->
<M> Generic Target Core Mod (TCM) and ConfigFS Infrastructure --->
[*] Fusion MPT device support --->
IEEE 1394 (FireWire) support --->
<M> I2O device support --->
[*] Macintosh device drivers --->
-*- Network device support --->
[*] ISDN support --->
<M> Telephony support --->
Input device support --->
Character devices --->
{M} I2C support --->
[*] SPI support --->
PPS support --->
PTP clock support --->
-*- GPIO Support --->
<M> Dallas's 1-wire support --->
{M} Power supply class support --->
{*} Hardware Monitoring support --->
{M} Generic Thermal sysfs driver --->
[*] Watchdog Timer Support --->
Sonics Silicon Backplane --->
Broadcom specific AMBA --->
<Select> < Exit > < Help >

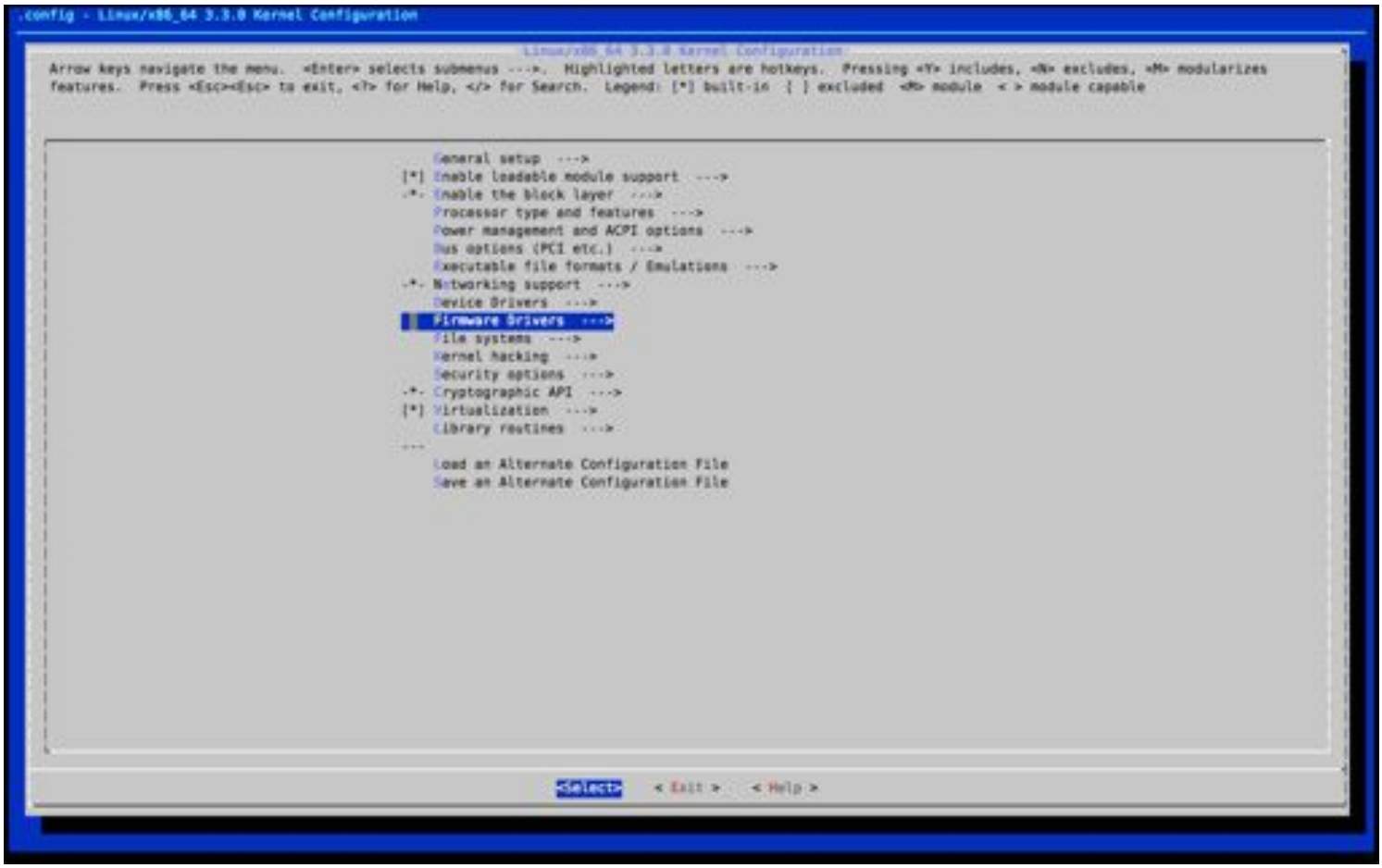

.config - Linux/x86_64 3.3.0 Kernel Configuration
Watchdog Timer Support
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<M> Eurotech CPU-1220/1410 Watchdog Timer
<M> IB700 SBC Watchdog Timer
<M> IBM Automatic Server Restart
<M> ICP Single Board Computer Watchdog Timer
<M> Intel 6300ESB Timer/Watchdog
<M> Intel TCO Timer/Watchdog
[*] Intel TCO Timer/Watchdog Specific Vendor Support
<M> IT8712F (Smart Guardian) Watchdog Timer
<M> IT87 Watchdog Timer
<M> HP ProLiant iLO2+ Hardware Watchdog Timer
[*] NMI decoding support for the HP ProLiant iLO2+ Hardware Watchdog Timer
<M> National Semiconductor PC87307/PC97307 (ala SC1200) Watchdog
<M> NS PC87413 watchdog
<M> nvidia TCO Timer/Watchdog
<M> SBC-60XX Watchdog Timer
<M> SBC8360 Watchdog Timer
<M> SMA CPU5 Watchdog
<M> SMSC SCH311X Watchdog Timer
<M> Winbond SMsC37B787 Watchdog Timer
< > VIA Watchdog Timer (NEW)
<M> W83627HF/W83627DHG Watchdog Timer
<M> W83697HF/W83697HG Watchdog Timer
<M> W83697UG/W83697UF Watchdog Timer
<M> W83877F (EMACS) Watchdog Timer
<M> W83977F (PCM-5335) Watchdog Timer
<M> ZF MachZ Watchdog
<M> Winsystems SBC EPX-C3 watchdog
<*> Xen Watchdog support
*** PCI-based Watchdog Cards ***
<M> Berkshire Products PCI-PC Watchdog
<M> PCI-WDT500/501 Watchdog timer
*** USB-based Watchdog Cards ***
<M> Berkshire Products USB-PC Watchdog
<Select> < Exit > < Help >


.config - Linux/x86_64 3.3.0 Kernel Configuration
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<M> Dallas's 1-wire support --->
{M} Power supply class support --->
{*} Hardware Monitoring support --->
{M} Generic Thermal sysfs driver --->
[*] Watchdog Timer Support --->
Sonics Silicon Backplane --->
Broadcom specific AMBA --->
Multifunction device drivers --->
[*] Voltage and Current Regulator Support --->
<M> Multimedia support --->
Graphics support --->
<M> Sound card support --->
[*] HID Devices --->
[*] USB support --->
<M> Ultra Wideband devices (EXPERIMENTAL) --->
<M> MMC/SD/SDIO card support --->
<M> Sony MemoryStick card support (EXPERIMENTAL) --->
-*- LED Support --->
[*] Accessibility support --->
<M> InfiniBand support --->
[*] EDAC (Error Detection And Correction) reporting --->
[*] Real Time Clock --->
[*] DMA Engine support --->
[ ] Auxiliary Display support --->
{M} Userspace I/O drivers --->
Virtio drivers --->
Microsoft Hyper-V guest support --->
Xen driver support --->
[*] Staging drivers --->
[*] X86 Platform Specific Device Drivers --->
Hardware Spinlock drivers --->
[*] IOMMU Hardware Support --->
[ ] Virtualization drivers --->
<Select> < Exit > < Help >

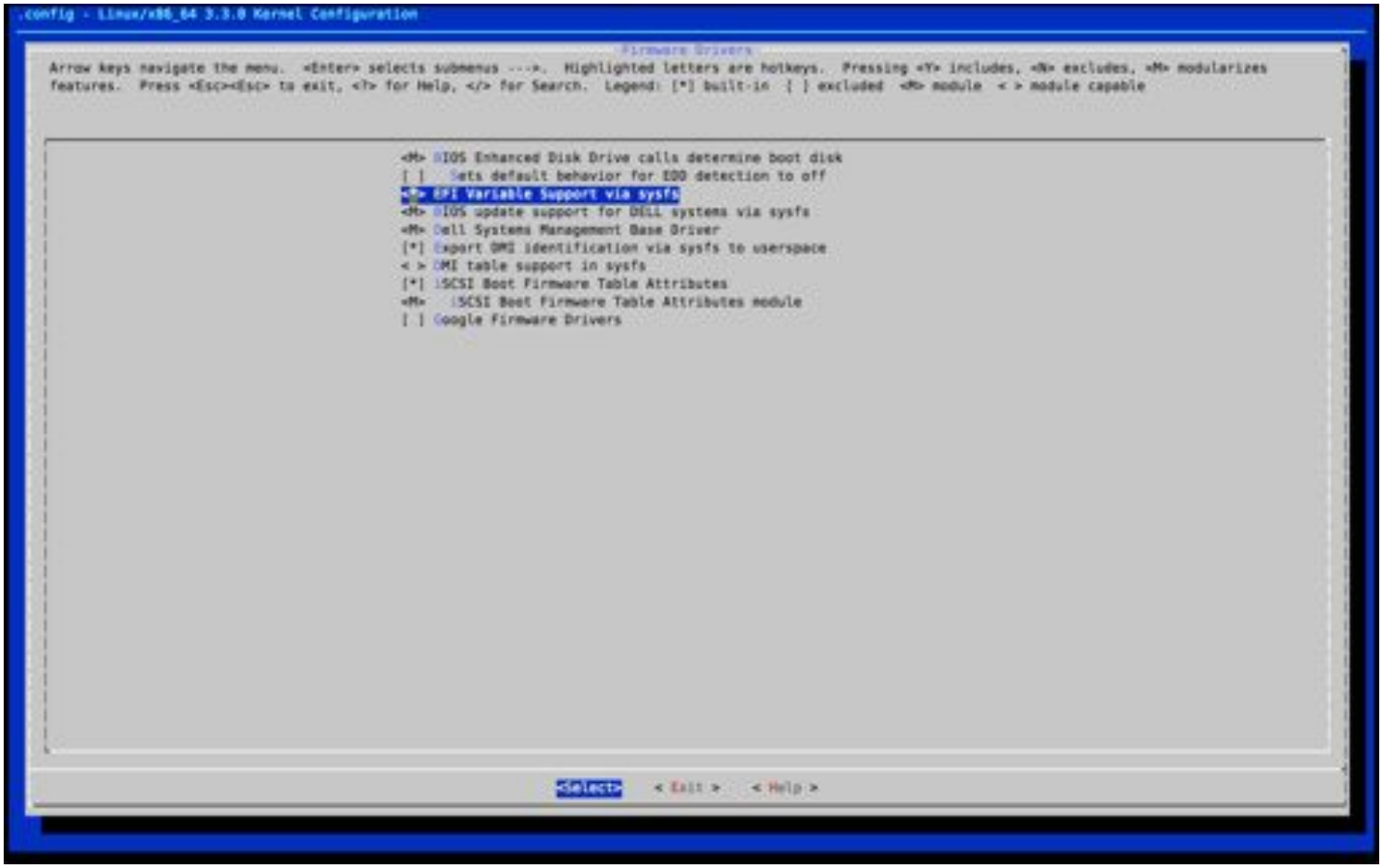

.config - Linux/x86_64 3.3.0 Kernel Configuration
Xen driver support
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] Xen memory balloon driver
[*] Memory hotplug support for Xen balloon driver
[*] Scrub pages before returning them to system
<*> Xen /dev/xen/evtchn device
[*] Backend driver support
<*> Xen filesystem
[*] Create compatibility mount point /proc/xen
[*] Create xen entries under /sys/hypervisor
<*> Userspace grant access device driver
<*> User-space grant reference allocator driver
<*> Xen PCI-device backend driver
<Select> < Exit > < Help >


.config - Linux/x86_64 3.3.0 Kernel Configuration
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<M> Dallas's 1-wire support --->
{M} Power supply class support --->
{*} Hardware Monitoring support --->
{M} Generic Thermal sysfs driver --->
[*] Watchdog Timer Support --->
Sonics Silicon Backplane --->
Broadcom specific AMBA --->
Multifunction device drivers --->
[*] Voltage and Current Regulator Support --->
<M> Multimedia support --->
Graphics support --->
<M> Sound card support --->
[*] HID Devices --->
[*] USB support --->
<M> Ultra Wideband devices (EXPERIMENTAL) --->
<M> MMC/SD/SDIO card support --->
<M> Sony MemoryStick card support (EXPERIMENTAL) --->
-*- LED Support --->
[*] Accessibility support --->
<M> InfiniBand support --->
[*] EDAC (Error Detection And Correction) reporting --->
[*] Real Time Clock --->
[*] DMA Engine support --->
[ ] Auxiliary Display support --->
{M} Userspace I/O drivers --->
Virtio drivers --->
Microsoft Hyper-V guest support --->
Xen driver support --->
[*] Staging drivers --->
[*] X86 Platform Specific Device Drivers --->
Hardware Spinlock drivers --->
[*] IOMMU Hardware Support --->
[ ] Virtualization drivers --->
<Select> < Exit > < Help >


.config - Linux/x86_64 3.3.0 Kernel Configuration
IOMMU Hardware Support
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
--- IOMMU Hardware Support
[*] AMD IOMMU support
[ ] Export AMD IOMMU statistics to debugfs
< > AMD IOMMU Version 2 driver (EXPERIMENTAL) (NEW)
[*] Support for Intel IOMMU using DMA Remapping Devices
[*] Enable Intel DMA Remapping Devices by default
[*] Support for Interrupt Remapping (EXPERIMENTAL)
<Select> < Exit > < Help >


.config - Linux/x86_64 3.3.0 Kernel Configuration
Linux/x86_64 3.3.0 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
General setup --->
[*] Enable loadable module support --->
-*- Enable the block layer --->
Processor type and features --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Executable file formats / Emulations --->
-*- Networking support --->
Device Drivers --->
Firmware Drivers --->
File systems --->
Kernel hacking --->
Security options --->
-*- Cryptographic API --->
[*] Virtualization --->
Library routines --->

Load an Alternate Configuration File
Save an Alternate Configuration File
<Select> < Exit > < Help >


.config - Linux/x86_64 3.3.0 Kernel Configuration
Firmware Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<M> BIOS Enhanced Disk Drive calls determine boot disk
[ ] Sets default behavior for EDD detection to off
<*> EFI Variable Support via sysfs
<M> BIOS update support for DELL systems via sysfs
<M> Dell Systems Management Base Driver
[*] Export DMI identification via sysfs to userspace
< > DMI table support in sysfs
[*] iSCSI Boot Firmware Table Attributes
<M> iSCSI Boot Firmware Table Attributes module
[ ] Google Firmware Drivers
<Select> < Exit > < Help >

```
root@xen:/home/src/linux-3.3# make-kpkg clean
exec make kpkg_version=12.036+nmu2 -f /usr/share/kernel-package/ruleset/minimal.mk clean
====== making target minimal_clean [new prereqs: ]======
This is kernel package version 12.036+nmu2.
test ! -f .config || cp -pf .config config.precious
test ! -e stamp-building || rm -f stamp-building
test ! -f Makefile || \
            make    ARCH=x86_64 distclean
make[1]: Entering directory `/home/src/linux-3.3'
  CLEAN   scripts/basic
  CLEAN   scripts/kconfig
  CLEAN   include/config include/generated
  CLEAN   .config
make[1]: Leaving directory `/home/src/linux-3.3'
test ! -f config.precious || mv -f config.precious .config
rm -f modules/modversions.h modules/ksyms.ver scripts/cramfs/cramfsck scripts/cramfs/mkcramfs
root@xen:/home/src/linux-3.3# fakeroot make-kpkg --initrd --revision=2.0.custom kernel_image
```

```
                      -plinux-image-3.3.0 -P/home/src/linux-3.3/debian/linux-image-3.3.0/
create_md5sums_fn () { cd $1 ; find . -type f | -regex './DEBIAN/.*' | -regex './var/.*'      -printf '%P\0' | xargs -r0 md5sum > DEBIAN/md5sums ; if [ -z "DEBIAN/md5s
ums" ] ; then rm -f "DEBIAN/md5sums" ; fi ; } ; create_md5sums_fn             /home/src/linux-3.3/debian/linux-image-3.3.0
chmod -R og=rX                /home/src/linux-3.3/debian/linux-image-3.3.0
chown -R root:root            /home/src/linux-3.3/debian/linux-image-3.3.0
dpkg --build                  /home/src/linux-3.3/debian/linux-image-3.3.0 ..
dpkg-deb: building package `linux-image-3.3.0' in `../linux-image-3.3.0_2.0.custom_amd64.deb'.
make[2]: Leaving directory `/home/src/linux-3.3'
make[1]: Leaving directory `/home/src/linux-3.3'
root@xen:/home/src/linux-3.3# df
Filesystem            1K-blocks    Used Available Use% Mounted on
rootfs                  7786092 1072948   6322600  15% /
udev                    6017760       8   6017752   0% /dev
tmpfs                   1204864     396   1204468   1% /run
/dev/mapper/xen-linux   7786092 1072948   6322600  15% /
tmpfs                      5120       0      5120   0% /run/lock
tmpfs                   2409720       0   2409720   0% /tmp
tmpfs                   2409720       0   2409720   0% /run/shm
/dev/sda2                251358   30289    208374  13% /boot
/dev/sda1                252900     128    252780   1% /boot/efi
/dev/mapper/xen-user   10452940 7275232   2653420  74% /home
root@xen:/home/src/linux-3.3# ls
arch     CREDITS  Documentation  fs       ipc      kernel       Makefile         modules.order   README          scripts   System.map  virt
block    crypto   drivers        include  Kbuild   lib          mm               Module.symvers  REPORTING-BUGS  security  tools       vmlinux
COPYING  debian   firmware       init     Kconfig  MAINTAINERS  modules.builtin  net             samples         sound     usr         vmlinux.o
root@xen:/home/src/linux-3.3# cd ..
root@xen:/home/src# ls
linux-3.3  linux-3.3.tar.bz2  linux-image-3.3.0_2.0.custom_amd64.deb
root@xen:/home/src# dpkg -i linux-image-3.3.0_2.0.custom_amd64.deb
Selecting previously unselected package linux-image-3.3.0.
(Reading database ... 31372 files and directories currently installed.)
Unpacking linux-image-3.3.0 (from linux-image-3.3.0_2.0.custom_amd64.deb) ...
Done.
Setting up linux-image-3.3.0 (2.0.custom) ...
Running depmod.
Examining /etc/kernel/postinst.d.
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 3.3.0 /boot/vmlinuz-3.3.0
update-initramfs: Generating /boot/initrd.img-3.3.0
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 3.3.0 /boot/vmlinuz-3.3.0
Generating grub.cfg ...
Found linux image: /boot/vmlinuz-3.3.0
Found initrd image: /boot/initrd.img-3.3.0
Found linux image: /boot/vmlinuz-3.2.0-1-amd64
Found initrd image: /boot/initrd.img-3.2.0-1-amd64
Found Debian GNU/Linux (wheezy/sid) on /dev/mapper/xen-b1
done
root@xen:/home/src#
```

Now that we have configured out kernel, we are ready to compile!

```
make-kpkg clean
fakeroot make-kpkg --initrd --revision=2.1.custom kernel_image
```

Compiling the linux kernel on my CPU took about 15 minutes, so feel free to take a breather.

Once the compiling has completed, provided you see no errors and a message about a package ending with .deb, you are ready to install your custom kernel!

```
cd ..
dpkg -i linux-image-3.3.0_2.0.custom_amd64.deb
```

The .deb file is portable and not 6 Gigabytes in size, so your modified kernel can be kept on a backup drive if you want it for future installations.

Known Bugs

My first tests was with Kernel 3.3, and it worked great, but when I tried re-producing these results Kernel 3.3.1 had been released. Kernel 3.3.1 with my instructions for some reason had kernel panics when booting with Xen, my advice is to stick to 3.3 for now.

Compiling Xen

Downloading a Specific Revision

At the time of testing I was using Xen 4.2 Unstable revision 25138, the current latest.

Throughout my testing I have compiled four previous revisions, and Xen is almost constantly improving, but occasionally an update will introduce a new bug. I suggest checking xenbits.xensource.com for the latest revision.

As before, we want to make sure our required packages have been installed:

```
sudo aptitude install grub-efi-amd64 sudo ssh bridge-utils parted ntfsprogs bzip2
build-essential libncurses-dev kernel-package fakeroot python-dev uuid-dev libglib2.0-dev
libyajl-dev bcc gcc-multilib iasl libpci-dev mercurial
```

We can now download Xen using mercurial, note that the compiled source will require just over 1 Gigabyte of space so be aware of where you put it. I borrowed the following lines from David Techer's blog.

```
rev=25138
hg clone -r $rev http://xenbits.xen.org/hg/xen-unstable.hg/ xen-unstable.hg-rev-${rev}
```

EFI Source Modifications

If you are using EFI you may want to modify the source before compiling. My experience has been that EFI Bootloader will interfere with Xen's memory mapping, and instead of seeing all of your RAM it'll only see 511 Megabytes. To fix this modifications have to be made to "xen/arch/x86/setup.c", search for the line containing "e820_raw_nr != 0" and apply these changes:

```
#if 0
    else if ( e820_raw_nr != 0 )
    {
        memmap_type = "Xen-e820";
    }
    else if ( bootsym(lowmem_kb) )
    {
        memmap_type = "Xen-e801";
        e820_raw[0].addr = 0;
        e820_raw[0].size = bootsym(lowmem_kb) << 10;
        e820_raw[0].type = E820_RAM;
        e820_raw[1].addr = 0x100000;
        e820_raw[1].size = bootsym(highmem_kb) << 10;
        e820_raw[1].type = E820_RAM;
        e820_raw_nr = 2;
    }
#endif
```

Compiling Process

The compiling process is actually quite simple, although the required package listing isn't really that clear which I believe creates most of the problems.

Run these commands and if you don't see any error messages a .deb file will be created that you can use to install Xen, and like with Linux it is portable:

```
./configure
make xen && make tools && make stubdom && make deb
```

Compiling took my system roughly 20 minutes, but I did not specify any concurrency flags.

If you want to run the installation of each item instead of a .deb that works fine as well, but I prefer the .deb as it is

portable, and the current Xen source creates it for you with the listed command. So now we navigate to the deb file and install it:

```
cd dist/
dpkg -i xen-upstream-4.2-unstable.deb
```

Post-Install Tuning

Once the installation runs we aren't yet ready to boot into Xen, we have to make some changes to get everything ready.

For our first step, we need to set some of the new init.d files to load at boot time using update-rc.d:

```
update-rc.d xencommons defaults 19 18
update-rc.d xend defaults 20 21
update-rc.d xendomains defaults 21 20
update-rc.d xen-watchdog defaults 22 23
```

You may ignore any warnings you see from running the above commands.

Next, if you visit your "/boot" directory, you will notice a bunch of files have been added. Most of these are not needed, and deleting them will reduce the clutter of your grub file later.

A simple "ls -l" command will reveal that 3 of the new .gz files are links to the original, and the xen-syms file is somewhat of a mystery. A very outdated source states that the xen-syms file contains debugging "symbols". I delete all of the links and xen-syms, and my system appears to function just fine.

Now we need to tell Xen to load first, to do that we can rename a grub script:

```
mv /etc/grub.d/20_linux_xen /etc/grub.d/09_linux_xen
```

Now we can run update grub and it will add our Xen system to "/boot/grub/grub.cfg", which we can modify if desired:

```
update-grub
```

Here is a copy of my Grub files Xen lines:

```
menuentry 'Debian GNU/Linux, with Xen 4.2-unstable and Linux 3.3.0' --class debian --class
gnu-linux --class gnu --class os --class xen {
        insmod part_gpt
        insmod ext2
        set root='(hd0,gpt2)'
        search --no-floppy --fs-uuid --set=root 6e3139b8-6cc6-4fa7-95f2-10bb99e76da3
        echo    'Loading Xen 4.2-unstable ...'
        multiboot       /xen-4.2-unstable.gz placeholder dom0_mem=1024M
        echo    'Loading Linux 3.3.0 ...'
        module  /vmlinuz-3.3.0 placeholder root=/dev/mapper/xen-linux ro quiet
xen-pciback.hide=(00:1a.0)(00:1b.0)(00:1d.0)(01:00.0)(01:00.1)(02:00.0)(03:00.0)
        echo    'Loading initial ramdisk ...'
        module  /initrd.img-3.3.0
}
menuentry 'Debian GNU/Linux, with Xen 4.2-unstable and Linux 3.3.0 (recovery mode)' --class
debian --class gnu-linux --class gnu --class os --class xen {
        insmod part_gpt
        insmod ext2
```

```
        set root='(hd0,gpt2)'
        search --no-floppy --fs-uuid --set=root 6e3139b8-6cc6-4fa7-95f2-10bb99e76da3
        echo    'Loading Xen 4.2-unstable ...'
        multiboot       /xen-4.2-unstable.gz placeholder
        echo    'Loading Linux 3.3.0 ...'
        module  /vmlinuz-3.3.0 placeholder root=/dev/mapper/xen-linux ro single
        echo    'Loading initial ramdisk ...'
        module  /initrd.img-3.3.0
}
```

Note that I used the dom0_mem flag to set my Dom0 memory to 1GB, and hid a variety of PCI devices. For now I recommend omitting these lines until I have a chance to explain them.

As this is a protected system file, if editing with vi you will need to use "wq!" to write and close.

Now we can restart and if all goes well we will see Xen at the top of our grub menu, and be greeted by the Linux login screen.

```
shutdown -r "now"
```

Once you login, we need to test whether our default Xen toolstack "xl" is working. The toolstack basically allows our privileged virtual machine (aka Dom0) to communicate with the Xen Hypervisor. In one step we can verify that it works and get some helpful information from our system:

```
xl dmesg
```

If this gave you an error, you may have missed a step and my only advice is to start googling. If it works, you'll get a verbose printout regarding Xen's status.

If you are running an EFI Bootloaded system, you will want to check your System RAM value to make sure all of it is there:

```
xl dmesg | grep -i "system ram"
```

If you see all of you RAM and xl is working, then congratulations your Xen compilation & installation was successful and we can move onto the last stage, installing a Windows HVM and passing your graphics card.

Configuring Xen for VGA Passthrough:

Alright, now that we have booted into Xen and verified the xl command is working, and that our memory is the right amount, we can begin setting up for an HVM installation.

My success with VGA Passthrough was as the secondary graphics device only, you will need some form of alternative connection initially to install the base operating system.

If you plan on running various tests like I did, you may consider creating a second LVM so you can store windows application and driver installers in case you need to reinstall. I found this saved me a lot of time.

Identifying PCI Devices in Linux

Most of the guides I found did not explain this, and as a Linux user who had only previously configured servers I had never tinkered with how Linux addresses hardware components.

To view PCI devices connected to linux, you can use the "lspci" command. Similarly for USB devices "lsusb". I didn't do any USB passing, so I will only be covering PCI in this section.

Some warnings to start with. If your board has a PCI Switch (most do) changing connecting devices can change the device identification and cause all kinds of confusion late in the game. In that same sense, adding new PCI devices to your HVM configuration in any order can affect its identification and render your system temporarily inaccessible. My suggestion is similar to measure twice cut once. I have not yet figured out how to undo pciback hidden devices, if anyone has I would love an addition. Another warning, passing one part of a device with multiple functions will often pass both devices.

In any event, PCI devices are referenced numerically by linux, it uses a colin and period to separate three main sections into this format "xx:yy.z", where xx is the bus, yy is the device, and z is the function. Here is the output for my lspci:

```
> lspci
00:00.0 Host bridge: Intel Corporation 2nd Generation Core Processor Family DRAM Controller
(rev 09)
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI
Express Root Port (rev 09)
00:02.0 VGA compatible controller: Intel Corporation 2nd Generation Core Processor Family
Integrated Graphics Controller (rev 09)
00:16.0 Communication controller: Intel Corporation 6 Series/C200 Series Chipset Family MEI
Controller #1 (rev 04)
00:1a.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset Family USB Enhanced Host
Controller #2 (rev 05)
00:1b.0 Audio device: Intel Corporation 6 Series/C200 Series Chipset Family High Definition
Audio Controller (rev 05)
00:1c.0 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port
1 (rev b5)
00:1c.5 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port
6 (rev b5)
00:1c.6 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port
7 (rev b5)
00:1c.7 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port
8 (rev b5)
00:1d.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset Family USB Enhanced Host
Controller #1 (rev 05)
00:1f.0 ISA bridge: Intel Corporation Z68 Express Chipset Family LPC Controller (rev 05)
00:1f.2 SATA controller: Intel Corporation 6 Series/C200 Series Chipset Family 6 port SATA AHCI
Controller (rev 05)
00:1f.3 SMBus: Intel Corporation 6 Series/C200 Series Chipset Family SMBus Controller (rev 05)
01:00.0 VGA compatible controller: ATI Technologies Inc Barts XT [ATI Radeon HD 6800 Series]
01:00.1 Audio device: ATI Technologies Inc Barts HDMI Audio [Radeon HD 6800 Series]
02:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection
03:00.0 USB controller: Etron Technology, Inc. EJ168 USB 3.0 Host Controller (rev 01)
04:00.0 USB controller: Etron Technology, Inc. EJ168 USB 3.0 Host Controller (rev 01)
05:00.0 PCI bridge: PLX Technology, Inc. PEX 8608 8-lane, 8-Port PCI Express Gen 2 (5.0 GT/s)
Switch (rev ba)
06:01.0 PCI bridge: PLX Technology, Inc. PEX 8608 8-lane, 8-Port PCI Express Gen 2 (5.0 GT/s)
Switch (rev ba)
06:04.0 PCI bridge: PLX Technology, Inc. PEX 8608 8-lane, 8-Port PCI Express Gen 2 (5.0 GT/s)
Switch (rev ba)
06:05.0 PCI bridge: PLX Technology, Inc. PEX 8608 8-lane, 8-Port PCI Express Gen 2 (5.0 GT/s)
Switch (rev ba)
06:06.0 PCI bridge: PLX Technology, Inc. PEX 8608 8-lane, 8-Port PCI Express Gen 2 (5.0 GT/s)
Switch (rev ba)
06:07.0 PCI bridge: PLX Technology, Inc. PEX 8608 8-lane, 8-Port PCI Express Gen 2 (5.0 GT/s)
Switch (rev ba)
06:08.0 PCI bridge: PLX Technology, Inc. PEX 8608 8-lane, 8-Port PCI Express Gen 2 (5.0 GT/s)
Switch (rev ba)
06:09.0 PCI bridge: PLX Technology, Inc. PEX 8608 8-lane, 8-Port PCI Express Gen 2 (5.0 GT/s)
Switch (rev ba)
08:00.0 PCI bridge: ASMedia Technology Inc. ASM108x PCIe to PCI Bridge Controller (rev 01)
0a:00.0 FireWire (IEEE 1394): VIA Technologies, Inc. VT6315 Series Firewire Controller (rev 01)
0b:00.0 Ethernet controller: Broadcom Corporation NetLink BCM57781 Gigabit Ethernet PCIe (rev
10)
```

You will notice that my graphics card is 01:00.0, but also 01:00.1, which is the onboard audio "function" of the

same device.

So, now you know that Linux identifies PCI devices using a numeric format, and that is how we can reference them. For getting further details on your devices check the lspci man pages.

VNC Console Configuration

Grub Configuration

Once we have identified the device names, we get to modify our grub.cfg file. If you want you can modify the scripts, but I find that is a bit harder to read, so I generally don't bother.

To hide PCI devices from Dom0 using your Grub configuration, we use the xen-pciback module, which should be a part of your custom compiled kernel.

We use the PCI identification in the format of ##:##.#, in a line as follows:

```
xen-pciback.hide=(##:##.#)
```

Subsequent devices are added with a new set of parenthesis:

```
xen-pciback.hide=(##:##.#)(##:##.#)
```

This line would be appended to the end of the module line in your grub.cfg. As an example here is my grub.cfg entry:

```
menuentry 'Debian GNU/Linux, with Xen 4.2-unstable and Linux 3.3.0' --class debian --class
gnu-linux --class gnu --class os --class xen {
        insmod part_gpt
        insmod ext2
        set root='(hd0,gpt2)'
        search --no-floppy --fs-uuid --set=root 6e3139b8-6cc6-4fa7-95f2-10bb99e76da3
        echo    'Loading Xen 4.2-unstable ...'
        multiboot       /xen-4.2-unstable.gz placeholder dom0_mem=1024M
        echo    'Loading Linux 3.3.0 ...'
        module  /vmlinuz-3.3.0 placeholder root=/dev/mapper/xen-linux ro quiet
xen-pciback.hide=(00:1a.0)(00:1b.0)(00:1d.0)(01:00.0)(01:00.1)(02:00.0)(03:00.0)
        echo    'Loading initial ramdisk ...'
        module  /initrd.img-3.3.0
}
```

A device hidden from Dom0 can still be found using lspci, but the driver will not be active, and the device is not controlled by Dom0. For this reason if you only have one graphics device, hiding it will effectively remove video from your Dom0.

HVM Configuration

First we want to choose an initial size for your Windows hard drive. I went with 40 Gigabytes, it was plenty of space to install and add some basic software for testing purposes. We will make a new Logical Volume which we can then add Windows to:

```
lvcreate -L 40G -n windows xen
```

That line will create a 40 Gigabyte volume named "windows" in the Volume Group named "xen". Obviously it won't work if you don't have available space or named your volume group differently.

Next we need to create a Windows configuration file, here is what mine looks like:

```
name='windows'
builder='hvm'
vcpus=4
maxvcpus=6
memory=6144
disk=[
                '/dev/xen/windows,,hda,w',
                '/dev/nas/software,,hdb,w'
        ]
vif=[
                'bridge=xenbr0,model=e1000'
        ]
pci=[
        '00:1a.0',
        '00:1b.0',
        '00:1d.0',
        '01:00.0',
        '01:00.1'
        ]
boot='c'
pae=1
nx=1
nestedhvm=1
viridian=1
videoram=16
stdvga=1
vnc=1
vncunused=1
vnclisten="0.0.0.0:10"
vncpasswd="password"
usb=1
usbdevice="tablet"
device_model_version="qemu-xen-traditional"
```

Depending on the name of your network bridge, and LVM partition you may want to modify those lines. Additionally the PCI devices will need to be modified to reflect your system.

You will want to get a copy of your Windows installation CD in iso format, and place it somewhere on your drive. To make a copy from a DVD insert the disk, verify the name using "parted -l" (it will be whichever device complains about being read-only), and run this command with appropriate adjustments to "if" (input file):

```
dd if=/dev/dvd of=/home/installations/win7.iso
```

Since the first boot needs to install Windows, we want to make a few minor adjustments to this configuration:

```
disk=[
                '/dev/xen/windows,,hda,w',
#                '/dev/nas/software,,hdb,w'
                '/home/installers/win7.iso,,hdc,r,devtype=cdrom'
        ]
vif=[
                'bridge=xenbr0,model=e1000,mac=5a:50:a3:14:b1:1c'
        ]
#pci=[
#        '00:1a.0',
#        '00:1b.0',
#        '00:1d.0',
#        '01:00.0',
#        '01:00.1'
#        ]
boot='dc'
```

I recommend commenting out the PCI devices until the system is installed and updated.

To start an HVM you use this syntax from termnal:

```
xl create windows
```

Where "windows" is the file containing your configuration, if it is in the current directory the name alone is fine, otherwise you want to provide the path.

After running the "xl create" command, provided you get no errors, you can verify that the machine is booted using:

```
xl list
```

You can monitor in more detail using this command:

```
xl top
```

Next we want to use VNC to connect to the HVM. As previously explained you will use your VNC Console utility, on OS X you can open it from terminal:

```
open vnc://10.0.1.20:5910
```

Substitute your Xen systems IP for mine, and if you used another display port then you will want to adjust that as well.

The installation window that appears in the VNC Console will like installing Windows normally. Once Windows is installed, I recommend getting all the Windows updates installed possible.

Some additional commands from terminal to help you control your system if it locks up:

```
xl destroy windows
xl reboot windows
```

My experience with VNC Consoles are that they can sometimes get laggy. You can close and re-launch the VNC Console without affecting the running system, so try that before resorting to xl commands.

Installing ATI Drivers

When you install any drivers, be they ATI or Windows default VGA Drivers, the VNC Console will go blank and the GPU will take-over. Do not be alarmed, this is normal. If you have not passed USB Controllers to the HVM you can still control the machine from VNC but you will need to use the connected monitor to see what you are doing.

My tests have been successful with versions 12.2 and 12.3 of the ATI Drivers. However, you may need to run the installer twice for complete functionality.

First, download the drivers from AMD's website. Then when you run the installer select "Custom":

```
Screenshot!!!
```

Be sure to deselect any of the unnecessary features, specifically ATI Catalyst Control Center (CCC). We only want the very basic set of contents, the drivers and required install items:

```
Screenshot!!!
```

Then you will need to reboot twice to make sure the installation worked successfully. Finally you can re-run the installation again, this time either express or custom with all check boxes should add ATI CCC successfully to the system:

```
Screenshot!!!
```

After that you will have a fully functional VGA Passthrough Windows 7 HVM! Congratulations you finished the guide! If you are curious about any other topics, I have prepared plenty of reference links below for your reading pleasure.

Known Bugs

A windows installation bug some may be aware of causes an enormous delay if you do not specifically format the partition before selecting to install. I have encountered this many times, so out of habit I select "new" and after it makes the partition I select "format" then "install".

ATI Drivers if installed with the express option may result in a BSOD, if that happens you can try booting into safe mode and removing the drivers. My success with driver removal was erratic, sometimes it worked other times I had to reinstall Windows.

The newer qemu-xen (not traditional) has better performance in most areas from my tests, but PCI Passthrough fails with this error per device:

```
libxl: error: libxl_qmp.c:239:qmp_handle_error_response: received an error message from QMP
server: Parameter 'driver' expects a driver name
```

The devices are listed then under xl pci-list windows, however when the machine shuts down xl logs (not qemu) indicates the devices were not actually passed with this error per device:

```
ibxl: error: libxl_qmp.c:239:qmp_handle_error_response: received an error message from QMP
server: Device 'pci-pt-00_1a.0' not found
```

Qemu Upstream has various bugs, it always logs a request to /etc/qemu-ifdown, a script or file that does not exist, and restarting HVM's fail, the qemu log clearly indicates the difference between shutdown and restart, but the xl logs always show destroy codes.

Not really a bug, but I would like a way to specify a domain id in the configuration, clearly it is being passed to qemu when launched, so if there isn't a flag available yet I want to see about adding one to the source. If anyone knows about this, please add it.

Other Related Topics:

While many guides for these utilities exist, I am writing these with specific Xen uses in mind.

SSH

SSH connections allow you to create a secure connection for SFTP file transfers and terminal access of a remote system. In our case it is extremely helpful for connecting to a Xen server from a remote machine. As an example, inside the network I can use my laptop from upstairs to start Windows, or reset the PFSense router.

Connecting through SSH is extremely simple. If you followed my guide, you should have it installed, and possibly even modified the default port. Default port modification is entirely optional, but it is a secure decision.

From Linux you can install SSH the same as was done in my guide. If using Mac it should already be included. If using windows Download PuTTY, it's a very tiny portable utility.

On Linux or Mac, here are some methods to connect over SSH:

```
ssh user@192.168.1.6
ssh -p 32456 user@192.168.1.6
```

The first line connects with the username of "user" to the machine at IP "192.168.1.6". The second specifies the port to use, this is only required if you changed the default port.

The first connection may ask you to accept an SSL certificate, this SSL certificate was generated by the system when you installed SSH, say yes to connect.

Every time you connect you should be prompted for the password of the user you are attempting to login as.

That concludes our SSH overview. I highly recommend connecting over SSH during this guide, it'll make the installation easier, and you don't have to pipe output to "more" because you can scroll back through it.

LVM Partitioning

Logical Volume Manager in Linux is a wonderful and flexible tool, and allows you to create dynamic work areas on one or more drives.

Before we get to the LVM specific commands, I wanted to put together a quick overview of how partitioning works, because it can get very confusing.

To start with we have three important parts to get familiar with. A Partition Table, Partitions, and File Systems.

A partition table, such as msdos or gpt is how the disk stores information about its partitions.

Partitions are assigned space, the start and length are usually stored by the partition table.

File Systems are applied to a partition, allowing you a structure by which to store and retrieve data within a partition.

So what makes this so complex? Well, instead of creating a file system for your partition, you could create another partition table and more partitions inside of that. LVM does exactly this, and without a firm understanding of the system it gets really hard to figure out what's happening.

As a visual aid, here is a crude drawing:

```
Hand Drawn Diagram
```

My SSD configuration is depicted by this drawing. It has three physical partitions, EFI, BOOT, and LVM. EFI has a FAT32 File System. BOOT has an Ext4 File System. LVM is managed by the systems Logical Volume Manager.

You will notice that inside the LVM I have two partitions, one for Linux, another for Swap. Swap doesn't have a file system, but the Linux partition is ext4.

But there are three more logical volumes as well: windows, pfsense, and squeeze. Each of these contains a partition table, not a file system.

Notice that Windows partition table contains one partition for Windows, and probably msdos bootloader code in the start of the windows LVM partition. Because Windows thinks this is a hard drive, it creates a partition table,

NOT directly as a file system.

The same is true of both Linux and PFSense, who have created partitions as well.

With LVM's we can choose to resize the partition, and then from inside the HVM we can expand the file system to fill the new space. In that same sense we could also shrink it in some cases.

Here is another diagram depicting a dual-boot machine, one part Windows the other part Linux. Size isn't so important, but notice that we have a partition table, a boot partition that probably uses Grub to load either Windows or Linux, and a partiton for each system:

```
Hand Drawn Diagram
```

Turning this into an HVM would be very difficult, either the msdos boot code is at the start of the partition tables, or if grub is launching windows then we would have to somehow create a new launcher with our HVM LVM Partition. Generally to do this you would have to copy the entire disk, which means you need an even larger partition to store the data.

In this example there isn't really a clean solution, but this is just to demonstrate each component and why it is important. Now that we understand them, we can begin learning how to manage Logical Volumes!

Let's start with some of the basic utilities:

- vgdisplay - Displays Volume Groups
- lvdisplay - Displays Logical Volumes
- lvcreate - Create a logical volume
- lvremove - Delete a logical volume
- lvresize - Resize a logical volume
- parted - GNU Parted utility used for modern partition management
- mkfs - Command used to create a file system when pointed at a partition.

Note that resizing the partition does not automatically tell the file system it can use all the newly created space. If you resize the LVM partition of your Windows HVM, you need to access Windows and expand the NTFS File System from the Windows Disk Manager.

The syntax to create a simple volume is as follows:

```
lvcreate -L 40G -n volume group
```

The syntax to delete a volume is:

```
lvremove /dev/group/volume
```

The syntax to resize a volume is:

```
lvresize -L+4 /dev/group/volume
```

Note that in all of the above examples, group is the name you gave the volume group, and volume is the name you gave the logical volume. In the create example "-L 40G" specifies a 40 Gigabyte partition size. In the lvresize "-L+4" says to add 4 Gigabytes to the partition.

I won't be writing a guide on parted, it's a bit complex. I recommend reading the man pages, just know that if you want to manage a partition you would probably use parted to do so.

If you created a logical volume for storage, you can go strait to creating a file system using the mkfs command as

follows:

```
mkfs -t type /dev/group/volume
```

Note that the "-t type" is where you specify the type of file system, such as ext4, ext3, ntfs, fat32, etc. To be able to create ntfs file systems you have to add the ntfsprogs package into your system. Similarly you need the dosfstools package for vfat, the newer FAT file system used by Windows 7.

That concludes a quick and dirty overview of LVM and Partitioning. I hope it helps with managing Xen HVM's.

Backup and Restore with dd

the dd utility is a fantastic tool that makes it possible to create backups and complete copies of your data, and at the same time has earned the nickname of "Data Destroyer" because it is easy to accidentally mixup the syntax.

To keep things strait, I remember the source and target as Input File, and Output File. While file isn't always accurate to our activity, it gets the general idea accross.

If I want to simply backup a whole drive to a file, I can do this:

```
dd if=/dev/sda of=/home/backup.img
```

For a single partition instead, I can add a number to "sda":

```
dd if=/dev/sda1 of=/home/backup.img
```

If I want to copy the contents of one drive to another I can do this:

```
dd if=/dev/sda of=/dev/sdb
```

With LVM's I can even make a new volume specifically for the purpose of backing up another volume:

```
lvcreate -L 40G -n backup xen
dd if=/dev/xen/windows of=/dev/xen/backup
```

Restoring is incredibly easy, simply reverse the input and output values. The only time this will not work is when you are attempting to overwrite the drive you are working from, if you need to restore your Linux system, you should boot into a Live CD to run a dd restore.

That concludes the dd tutorial.

References:

This is a list of referenced material during my testing, each of the items listed served a specific function in the creation of this guide.

For installing EFI Grub on Wheezy I found this short and simple guide to be exceptional:

http://blog.garyhawkins.me.uk/?p=185

This guide was very helpful and mostly accurate for compiling the Linux Kernel:

http://vanilja.org/kernel/

Kernel Source

http://kernel.org/

Kernel Flag Database, was super helpful in verifying the various kernel flags:

http://cateee.net/lkddb/

Compilation Instructions were somewhat invalid for 4.2 but contain some helpful steps and information:

http://wiki.xensource.com/xenwiki/Xen4.1

http://wiki.xensource.com/xenwiki/Xen4.0

Mercurial Xen Source:

http://xenbits.xen.org/hg/xen-unstable.hg/

David Techer's Blog has nVidia patch information, and I used his method to set the revision for mercurial download:

http://www.davidgis.fr/blog/index.php?2012/02/22/884-xen-42-the-way-to-compile-xen-has-changedrevision24869

EFI Bootloader memory error, this fellow encountered it and posted his solution, which I used with some adjustments for the latest source:

http://serverfault.com/questions/342109/xen-only-sees-512mb-of-system-ram-should-be-8gb-uefi-boot

This contains the manuals for commands and configurations of xl, xm, and various other Xen components:

http://wiki.xen.org/wiki/Xen_Man_Pages

The only location I found mention of xen-syms, clearly outdated:

http://xen.xensource.com/files/xen_user_manual.pdf

HTML to Wiki converter, used to turn this HTML built guide into a Wikipedia copy paste:

http://labs.seapine.com/htmltowiki.cgi

http://en.wikipedia.org/wiki/Wikipedia:Tools/Editing_tools#From_HTML