PA−RISC/Linux Boot HOWTO

Deb Richardson

The Puffin Group

deb@thepuffingroup.com

Thomas Marteau

ESIEE

marteaut@esiee.fr

Revision History

Revision 0.9	January 15, 2002	Revised by: tm
This version brings you some useful advices for compiling your own kernel on hppa systems.		
Revision 0.8	October 17, 2001	Revised by: tm
This version takes care of the change of name of the official FTP and CVS sites and modify the license.		
Revision 0.7	October 13, 2001	Revised by: tm
This version adds some updates due to the progress of PA/Linux.		
Revision 0.6 draft	September 26, 2001	Revised by: tm
This version contains some minor changes and complete the "obtaining bootp/tftpd" section.		
Revision 0.5 draft	August 3, 2001	Revised by: tm
This version is a large update from Debbie's work		
Revision 0.3 draft	November 24, 1999	Revised by: dlr
The initial and published version of this HOWTO.		

This document outlines the procedures for getting the current PA−RISC/Linux development kernel to boot on your PA−RISC system. It also explains the functions of PALO, the kernel loader for PA/Linux. Also, you will find many informations about how to compile a kernel from the source available at http://cvs.parisc−linux.org/. Please notice that this HOWTO is a newer version than Deb Richardson's and includes more accurate informations due to the progress of the port. Nevertheless, I must say that this version keeps some texts of the oldest one and I reveal some hidden works of Deb.

If you are looking for some informations related to HP hardware but not directly to PA−RISC, please read Bruno Cornec's HP−HOWTO.

For more information about the PA−RISC/Linux porting project, please see http://www.parisc−linux.org/. This site deals with kernel development and improvment. For userspace troubles, please refer to Debian's port pages.

Table of Contents

1. Copyright and Licensing.....1

2. Supported Hardware.....2

3. Preparing to boot.....3
3.1. Consoles.....3
3.1.1. Using a graphic console.....3
3.1.2. Using a serial console.....3
3.2. Preparing to boot from the network.....6
3.2.1. rbootd or bootp?.....6
3.2.2. Using rbootd.....6
3.2.3. Using bootp/tftp.....7
3.2.4. Booting your PA−RISC/Linux system via network and with serial console.....8

4. Building your own PA−RISC/Linux kernel.....10
4.1. GCC compiler.....10
4.1.1. native build.....10
4.1.2. cross compiled build.....11
4.2. Kernel configuration.....11
4.3. Kernel installation.....11

5. Booting your PA−RISC/Linux system via PALO, the kernel loader.....13
5.1. What is PALO?.....13
5.2. What does PALO?.....13
5.3. How to make a lifimage with RAMDISK ?.....13
5.4. How to make a lifimage with NFSROOT ?.....13
5.5. How to make bootable a partition ?.....14
5.6. How to use PALO at the boot ?.....14
5.6.1. The theory.....14
5.6.2. One example.....15

6. What you need to know about BOOT_ADMIN.....17
6.1. The main commands.....17
6.2. The configuration commands.....17
6.3. The information commands.....17
6.4. The service commands.....18

7. HOWTO contributors.....19

1. Copyright and Licensing

2. Supported Hardware

Releasing the 0.9.3 version of PA/Linux, a great improvment was made in term of quantity and quality of hardware support. That's why, even if your model is not listed, give it a try and report your result to the mailing list. The following PA−RISC machines can be booted like any other box of a different architecture. I must add that this list can change at any time. The best way to get an updated list is to look at http://mkhppa1.esiee.fr/list.html. At this URL, you will know if your hardware is supoorted and how. If you can run PA/Linux *via* the serial console or *via* the graphic card. You will also find everything you need there like the PDC firmware updates.

- any 712 models. You can get X running on them via framebuffer device!
- any 715 models including Strider series.
- any 710, 720, 730, 750 models should be running with the latest ISO. It contains some modifications specially for hard disk devices.
- some 725, 735, 755 models are running with the latest kernels. But since there was not a lot of feedback about these machines, we can not be more explicit.
- A180C and similar.
- A500 and similar.
- BXXX models like B132, B180. These boxes can be used in the framebuffer mode *via* the Standard Text Interface.
- CXXX models like C110, C160.
- CXXXX models work with the I/O devices linked to the first IOMMU.
- D class is normally working. However, many very different configurations exist in the D−class. As a result, it is quite difficult to make a general statement.
- J class is the SMP version of CXXXX models. So, I have to do the same remark.
- L1000 and L2000.

The following hardware will work in the future:

- L3000 model.
- N class.
- SuperDome class.

The following hardware won't work:

- very old boxes like 705s.
- E,F,G,H,I class. It seems that Christoph Plattner made his E55 running.
- T,V class

3. Preparing to boot

Like any other system, machines based on PA−RISC processors has several steps in order to be running. First of all, you must set properly the hardware parameters. Then, it must get a first boot in order to launch the installation process. Once the kernel is set up, you can reboot and then you install all your file system. At this point, your system is installed and running. This paper tries to separate the text into these different moments.

This first section introduces several HP hardware parameters configurable *via* HP **BOOT_ADMIN**

3.1. Consoles

In order to boot your PA−RISC system with the PA/Linux kernel, you must first get a console on that system. You can use either a graphic console, which requires that a monitor be attached to the system, or a serial console, which allows communication with the system from a remote Linux machine. For the graphic console, you must be sure that the Linux support the graphic card of your system and there are two ways to get the graphic console. If you think about bug reporting your trouble, you must differenciate the two of them. First, the STI console is the classical video text console like on your PC for example. This name is due to the fact that each HP box has the Standard Text Interface which contains some standard ways to access to video memory. Second graphic console is the framebuffer console. In fact, while booting, you will see a penguin appeared on the top−left corner. This is the easiest way to differenciate the two graphic modes. For the serial console, it is a good way to get all the prompt messages including the BOOT_ADMIN ones. It is very useful for bug reports. The majority of servers can only be used by the serial console.

If you have a monitor for your PA−RISC system, the easiest method is to use a graphic console. If you have troubles, the best way is serial console.

3.1.1. Using a graphic console

Using a graphic console is just a fancy way of saying "use the monitor attached to the machine". If you have a monitor for your PA−RISC system, you should be able to bring up a console using the monitor, without having to mess with changing to and configuring a serial console.

Obviously, if you can use a graphic console, this is the easiest way to proceed. Nevertheless, you must be sure that your hardware is supported.

3.1.2. Using a serial console

The only time that you should have to use a serial console is if you either don't have a monitor for your PA−RISC machine, or your machine doesn't support graphics. Also, the kernel can NOT use the graphics console of some models, at the moment. So, if you need to use a serial console, use the following procedures to set up serial console support.

3.1.2.1. Serial Cable

To connect your PA−RISC machine to your PC so you can boot the system using a console, you need a 9 pin−to−9 pin female null−modem cable. You should be able to obtain such a cable at your local computer hardware supplier. Obviously, you can connect the other end of the cable to a terminal but the best is to connect it to another box running **minicom** to get every messages and to copy and paste them logged in a file in order to get a complete and easy bug report.

3.1.2.2. Configuring minicom on Linux

In order to communicate with your PA−RISC machine during this process, you have to set it up in "serial console" mode and configure a serial communication program. We recommend **minicom**,which is installed in most Linux distributions by default. If you don't have minicom on your system, you can find the latest package at any of the major Linux software sites.

Most of the minicom configuration is machine−dependent. You must, however, ensure that:

a. the baud rate is set to 9600
b. parity is set to 8−N−1

Don't worry too much because it is the default value for PA−RISC system. If you have a PC, you will probably need to change the baud rate but it seems that PA−RISC systems do like that too much.

3.1.2.3. Switching consoles

Before you can set up a serial connection between your Linux machine and your PA−RISC machine, you have to change the console mode on the PA−RISC system.

The default console mode is `graphic_1`. To change this, use the following procedure:

1. Turn the PA−RISC machine on.
2. During the boot process, the following message will appear:

   ```
   Searching for Potential Boot Devices.
   To terminate search, press and hold the ESCAPE key.
   ```

   When this message appears, press and hold the **Esc** key until an options menu appears.

3. By default, you enter the BOOT_ADMIN console. In some 715s, the options menu looks like this:

   ```
   b)    Boot from specified device
   s)    Search for bootable devices
   a)    Enter Boot Administration mode
   x)    Exit and continue boot sequence
   ?)    Help

   Select from menu:
   ```

   Select "a) Enter Boot Administration mode". This will bring up a "`BOOT_ADMIN>`" prompt. Everything else you do will be in BOOT_ADMIN mode. So now, everybody is in the BOOT_ADMIN console!

3.1.2.3.1. Checking current console mode

Type: **path console** to see to which mode the console is currently set.

If it's in graphic console mode, it will return "`Console path  = graphic_1`".

If it's set to serial console, it will return "`Console path  = rs232_a.9600.8.none`" or something similar.

Note that for some models, you can find some differences but the idea is the same. If you want to see more descriptions, please write a message telling me the box you use and what you get.

3.1.2.3.2. Changing to serial console mode

To change to serial console mode, type the following command at the BOOT_ADMIN command prompt:

path console rs232_a.9600.8.none

To verify that the console path has been properly set, type **path console**. This should return `Console path  = rs232_a.9600.8.none`, indicating that the system is now set to boot in serial console mode. By default, **reset** will make reboot your system with the new parameters.

3.1.2.3.3. How can I change the boot console to serial on a 712?

Unfortunately, you can't. Although 712s are configured for in−house HP development to use serial console, this configuration isn't supported in the field. You have to use a graphics console for 712s but if you really want it, you have a tip in the PA/Linux mailing list archive. In fact, why will we use this beautiful 712 with serial console when you can have X!

3.1.2.3.4. Changing to graphic console mode

It is the opposite operation compared to the upper one. By checking your console path, you should see `Console path  = rs232_a.9600.8.none` Now, you must set your grahic mode *via* **BOOT_ADMIN**

path console graphic_1

You should get the display available after a **reset**. If the screen does not seem to work, try to press the **Tab** key to change the resolution of your display. Pressing this key, the monitor type changes to one resolution to another one. Perhaps, you need to this operation several times. This is true also when you change your monitor.

3.2. Preparing to boot from the network

This is a very old way to operate but it used to be the only way. Now, you do not need to boot *via* the network except some very specific case. That's why it is still here.

3.2.1. rbootd or bootp?

All new machines can boot using **bootp**, including the 715/100, 715/120, as well as all 712s.

3.2.2. Using rbootd

3.2.2.1. Obtaining rbootd

If you have an older machine that requires **rbootd** to boot off a network, use the following procedure to set up, configure, and boot using the PA−RISC/Linux kernel.

Older machines, including the Scorpio 715s, require **rbootd**. You can obtain **rbootd**:

- for all the distributions except Debian, you need to get this archive *via* http://mkhppa1.esiee.fr/download.html
- for Debian, from http://www.debian.org/Packages/stable/net/rbootd.html

3.2.2.2. Configuring rbootd

For example, to boot a PA−RISC 715 system, you have to have rbootd installed on the Linux system (a "boot server") where you will be storing the PA−RISC/Linux kernel image that you want to use to boot your PA−RISC system.

Once installed, use the following procedure to configure **rbootd** to work with your PA−RISC system:

1. In `/etc/rbootd.conf` add the following line:

   ethernet addr *`bootfile`*

   Replace *`bootfile`* with the name of your PA−RISC/Linux kernel image, usually "`lifimage`".

2. Now get the ethernet address of your PA−RISC system by typing **lan_addr** at the BOOT_ADMIN prompt on your PA−RISC system.

   It will return a number like `080009−7004b6`. Make note of the number returned.

3. In `/etc/rbootd.conf` on your boot server, the ethernet address has to be colon−delimited. So, you'll have to modify the number you just obtained so every two characters (after removing the "−") is separated by a colon. For example:
   ```
   080009−7004b6
   ```

becomes

```
08:00:09:70:04:b6
```

Add the colon delimited ethernet address to `/etc/rbootd.conf` on your boot server. The resulting file will look something like this:

```
# ethernet addr        boot file(s)        comments

08:00:09:87:e4:8f      lifimage_715        # PA/Linux kernel for 715/33
08:00:09:70:04:b6      lifimage_720        # PA/Linux kernel for 720
```

This `rbootd.conf` example contains the ethernet addresses and boot file names for two different machines.

Once you have changed the configuration file, restart **rbootd**.

By default, **rbootd** assumes that bootfiles are located in `/var/lib/rbootd/`. If you use the archive for others distributions, this directory is `/usr/mdec/rbootd/`. Therefore, you will have to put your bootable kernel image in that directory, or, if you really hate that directory for some reason, you can recompile **rbootd** to use a different directory.

The easiest thing, of course, is to just drop your kernel images in your default directory!

3.2.3. Using bootp/tftp

3.2.3.1. Obtaining bootp/tftp

For Debian users, you just have to install the packages *via* these commands as **root**:

```
apt-get install bootp
apt-get install tftpd
```

> The package **bootp** can be replaced by **dhcp**. But, this howto won't go further in this way. For your information, after installing the package **dhcp**, you have a section in `/etc/dhcpd.conf` about BOOTP and you can always try **man dhcp**.

If you need rpm packages, the best advice is to go to http://rpmfind.net. It looks like for Red Hat users, you need to create the user "nobody" belonging to the group "nogroup". The files contained in your `/tftpboot/` directory should belong to him.

3.2.3.2. Configuring bootp/tftp

Use the following procedure to use **bootp** on your boot server:

This section is dedicated to Debian users. For others distributions, it is similar but there can be some changes like default directories. Since Debian maintained a **bootp** package, I will discuss about it only. For **dhcp** configuration, it is not planned to be added.

1. Configure `/etc/inetd.conf` on your boot server by adding the following lines:

```
tftp            dgram   udp     wait    nobody  /usr/sbin/tcpd       /usr/sbin/in.tftpd /tftp
bootps          dgram   udp     wait    root    /usr/sbin/bootpd     bootpd -i -t 120
```

   Here, `/boot` is being used for tftpd server. You can choose another directory if you want. According to **man tftpd**, the default directory is `/tftpboot`.

   When this is done, restart **inetd** with: **/etc/rc.d/init.d/inetd restart**.

2. According to **man bootptab**, set up the `/etc/bootptab` file to contain:

```
[hostname]:hd=/tftpboot/Image:\
                             :rp=/usr/src/parisc/:\
                             :ht=ethernet:\
                             :ha=[mac address]:\
                             :ip=[ip address]:\
                             :bf=[boot filename]:\
                             :sm=255.255.255.0:\
                             :to=7200:
```

   You have to fill in the *`[hostname]`*, *`[mac address]`*, and *`[ip address]`* with the appropriate information, of course, where:

   - [hostname] is the name of the PA−RISC host
   - [mac address] is the ethernet address of the PA−RISC box, which you obtain by typing **lan_address** at the `BOOT_ADMIN>` prompt
   - [ip address] is the IP address of the PA−RISC system
   - [boot file name] is the name of the bootable kernel image.

   You'll end up with something like this:

```
vodka:hd=/tftpboot:\
                             :rp=/usr/src/parisc/:\
                             :ht=ethernet:\
                             :ha=080069088717:\
                             :ip=140.244.9.208:\
                             :bf=lifimage:\
                             :sm=255.255.255.0:\
                             :to=7200:
```

3.2.4. Booting your PA−RISC/Linux system *via* network and with serial console

To conclude with the development way to boot the kernel, this section will tell you how to boot if you are using a server to boot your HP system. But it tends to less and less used. For users, please refer directly to **PALO** section.

Here we are. This is just some tips to get the boot for those who tried the network way. You've done everything outlined above, your development machine is hooked up to your PA−RISC machine, you've got a bootable PA−RISC/Linux kernel image on your boot server, and you're ready to give it a try. If everything is as it should be, the following procedure will allow you to boot your PA−RISC system into Linux.

1. Make sure your development machine is connected to your PA−RISC machine with a serial cable. Sounds obvious, but check anyway.
2. Fire up **minicom** on your development machine.
3. Start your PA−RISC system up.
4. Watch your minicom console. When the following message appears during the PA−RISC machine's boot process, press and hold the **Esc** key:

```
Searching for Potential Boot Devices.
To terminate search, press and hold the ESCAPE key.
```

5. Select "`a)  Enter Boot Administration mode`" from the menu. This brings up the `BOOT_ADMIN>` prompt.
6. Type the following at the prompt: **boot lan**.
7. Watch your PA−RISC system magically become a PA/Linux system. Ta dah!

---

4. Building your own PA–RISC/Linux kernel

To build a Linux kernel, you need a compiler and the Linux source. The first element is not a trivial thing to find because it depends on how you build your kernel. The second is easier since it lives on the official CVS site. First, we will discuss about **GCC** compiler. Then, the configuration of your build will be treated. The last paragraph will deal with the installation of this new kernel.

4.1. GCC compiler

You can compile your kernel with your own HP box. But on oldest systems, perhaps, you prefer to use another computer to compile your kernel. Let's see the two alternatives:

4.1.1. native build

Since Debian is the only distribution supporting PA–RISC architecture, if you want to use the *Super Cow* powers, you need to have some basic informations about Debian packages system first.

4.1.1.1. apt–get and friends

apt–get is a simple command line utility that manages your Debian package system. First, Gustavo Noronha Silva wrote APT HOWTO that I invite you to read for deeper knowledge. Here, we just want to build a kernel. Since hppa port is not out yet, you should be very careful with the mirrors you choose in your `/etc/apt/sources.list`. For example, In Germany, you can use the following settings:

```
# non-US packages
deb http://www.ftp.uni-erlangen.de/pub/debian/ unstable/non-US main non-free contrib
# Binary packages
deb http://gluck.debian.org/debian unstable main contrib non-free
# source packages
deb-src http://gluck.debian.org/debian unstable main contrib non-free
```

4.1.1.2. update your gcc

If you are using your own HP box, you need only the classical **GCC** compiler. The recommendation is to update to the last version uploaded by the developpers.

```
apt-get update
apt-get upgrade
```

If you do not want to upgrade all your system, the package description of kernel–source, you need to get those packages updated:

- binutils
- fileutils
- gcc
- libc–dev
- make

When you did this step, you can procede to the kernel tweaking.

4.1.2. cross compiled build

For this way of building your kernel, everything depends on the architecture of your building machine. For PC, you can download an already−made cross compiler on PA/Linux FTP server. For the others architectures or if you want to compile your own toolchain, please refer to Carlos O'Donell's HOWTO.

4.2. Kernel configuration

The best way to get performance is to get a well configured kernel. For PA−RISC platform, **make oldconfig** is a kind of default setup. If you want to make your own kernel, the first step is to know what is your hardware. The best way is to look at your box and find a maximum of information. Then, you go to the official hardware database or the HP partsurfer.

When you know what is inside your box and what you want to do with your box, just run **make menuconfig** or another config command. Here is the list of menus you should be going in to see if the value set corresponds with your hardware:

- *Processor type* indicates your CPU model
- *General options* tells you what is going to be enable in your kernel
- *Network device support* is used to set your network card
- *Character devices* shows your I/O possiblities
- *Console drivers* is directly related to your console path
- *Sound* enables your Harmony hardware

As you see, menus about HP hardware are not numerous but there is a lot of dependency between them. Now, you must configure accordingly to the use you will do of the box. Here is the list of some menus you should be going in to configure the services you want:

- *General setup* is responsible for binary formats
- *Parallel port support* gives you the choice to deactivate parallel port
- *Block devices* set on the ramdisk and loopback support. You probably won't use them.
- *File Systems/Network File Systems* is where to go to unset NFS support

Once this is done, save your configuration. Everything is written in the .config file. You can backup it because a **make distclean** will remove it. At this stage, you do **make dep vmlinux** and if everything goes fine, you have a new kernel.

4.3. Kernel installation

If you made a native build, you backup the last working kernel with an extension like ".bk" and you copy the recent kernel with the same name. You reboot and try the new one. If it is not working, you can reboot and using **PALO** command, you change the name of th kernel you want to boot. (See next section for more informations)

If you are booting *via* network, you need to set **PALO** as it is explained in the next section and run **make palo**

5. Booting your PA−RISC/Linux system *via* PALO, the kernel loader

5.1. What is PALO?

PALO is two programs, a boot loader, which is loaded by the HP firmware into memory and then executed, and boot media management tool, which initializes and updates bootable media such as disks. The PALO boot loader executable is stored in a file called iplboot. "IPL" is HP jargon for Initial Program Loader. The boot media management tool is called **PALO**, just as on x86 the lilo boot media management tool is called **LILO**, though it's worth noting that **PALO** doesn't usually need to be used every time you build a new kernel, as **LILO** does.

5.2. What does PALO?

The main idea is to boot a kernel passing all the parameters it needs. The practical experience is a little more complicated. Indeed, PALO can transform the classical vmlinux into a HP−UX bootable lifimage including a RAMDISK or a NFSROOT redirection. However, it could make a hard disk drive bootable specifying the console output and the root device. We are going to see all these points precisely. The major point is that lifimage file is a bootable kernel and vmlinux is the kernel itself and it needs **PALO** to be bootable.

5.3. How to make a lifimage with RAMDISK ?

First point is to explain when you should use this way. At an earlier step of PA/Linux project, the lifimage was very useful. In fact, you put this file in the bootp server tree and then you can boot your HP box *via* boot lan instruction. The advantage of ramdisk is to unpack its own file system and to be completely independant. The main drawback is the fact that you have to build your own ramdisk if you have some memory constraints or some files customized. Now, let's see how to get a lifimage. So, you get the latest source of PA/Linux. Mainly, you will need a good cross−compiler and the linux and palo directory. Everything you need is at http://www.parisc−linux.org/. You do your **make menuconfig**. Then, you just have to do **make palo**. The point is that in the palo directory, you have the **PALO** `Makefile` in which you have to select the ramdisk file. So, initially, you can read:

```
# RAMDISK = ${PA}/ramdisk.bin
```

Thus, if you want to use the **RAMDISK** support with a file called for example ramdisk.bin and placed in the linux directory, you just have to uncomment it:

```
RAMDISK = ${PA}/ramdisk.bin
```

After configuring the `Makefile`, you can go into the linux directory and launch your **make palo**. The result, a lifimage file, is waiting for you in the **PALO** directory.

5.4. How to make a lifimage with NFSROOT ?

This method is widely used because the kernel and the file system is visible since your server. It is also very easy to test a new kernel. You just have to generate the kernel and then you put it in the correct directory. After rebooting, the HP box will find *via* **boot lan** instruction its new kernel. Getting the **NFSROOT** support is easier than the **RAMDISK**. You edit the `Makefile` of **PALO** and you specify the tftpboot server IP adress. In fact, if your server has 10.10.10.2 for IP adress, then the default file is ok because we can read into:

```
NFSROOT = 10.10.10.2
```

If you have anything else, this field must be completed by the correct data. After configuring the `Makefile`, you can go into the linux directory and launch your **make palo**. The result, a lifimage file, is waiting for you in the **PALO** directory.

5.5. How to make bootable a partition ?

This part is where **PALO** will be seen as **LILO**. **PALO** is mainly a program that enables HP box to boot *via* a kernel on its own hard disk drive. This section is going to explain you how to make it right. When you install the **PALO** package, Paul Bame, the author and maintainer, put a copy of the default `/etc/palo.conf` in the `/usr/doc/share/doc/palo/palo.conf`. If you want to understand how **PALO** works, you just have to read this file! This sample comes from `palo.conf`. It is the default value. We advice you to stay close to this scheme. However, the parameter **recoverykernel** is the path to the kernel that you want to boot with in a failsafe session! The next one, **bootloader**, is the path to this utility which is produced by **PALO** when you ask for **make iplboot**. **init−partitioned** is used to tell on which device you want to write the result. The effect is immediate. It means that **PALO** is going to write on the first octets of this partition. A good advice will be to check this info with **fdisk**, in order to be sure to update the good drive. Finally, the last parameter is the **commandline**! The first number indicates which partition **PALO** has to mount in order to get the kernel file! Logically, the following string is the absolute path to the kernel. **HOME** and **TERM** seem to be some environmental parameters passed to **init**. They are not compulsory but they can be useful. The **root** parameter explains to the kernel which partition it must mount for the root file system in read−only while booting. It can be tricky when you plug two disks. You could add some very interesting parameters like **console** where you can specify the output for the console. You should know that **console=ttyS0** is for a serial console and **console=tty0** is for a STI−console. In the latest version of **PALO**, this path should be added automatically and correctly. If not, please mail to the mailing list.

```
# The following arguments are set up for booting from /dev/sda3, specifically
# mounting partition 3 as root, and using /boot/vmlinux as both the
# recovery kernel, and the default dynamically−booted kernel.
# −−recoverykernel=/boot/vmlinux
# −−bootloader=/boot/iplboot
# −−init−partitioned=/dev/sda
# −−commandline=3/boot/vmlinux HOME=/ TERM=linux root=/dev/sda3
```

5.6. How to use PALO at the boot ?

5.6.1. The theory

After installing your **PALO**, you would like to modify the parameters you pass to the kernel. First, you must know how to interact with **PALO** at the startup. For some old models, you must add ipl to your command in the boot admin console: **BOOT_ADMIN> boot pri ipl** For the recent HP boxes, the system will ask you if you want to interact with IPL. You just have to answer by a "y". Then, you are in **PALO** with the list of all the parameters and the corresponding number. Then, you enter the number of the parameter you want to change. You hit Enter and you modify it and you validate. The system will redisplay the new list. This modification is not permanent! To write your changes, you have to run **/sbin/palo** and it will write on the disk all the parameters contained in the default file, eg `/etc/palo.conf`. If you want to add another parameter, you select one parameter and write yours with a space between the two. If you want to delete one, you select it and erase the complete entry. You will see that the list counts one parameter less.

For more informations about **PALO**, please look at the PALO readme. This section is mainly inspired from this Paul Bame's file and my current page about **PALO** that you can find at http://mkhppa1.esiee.fr/palo.html.

5.6.2. One example

This example is the work of *Michael Damaschke*. So, let's go for the story of the happy PA/Linux user booting a kernel, also called *I don' t know how I can configure my workstation to boot the right kernel I want ?*.

After the power on of your workstation and monitor there is a message on the screen how told you that the workstation wants to start automatically an bootdevice or you can hold the **Esc** key to break up the auto−booting., on this situation you must hold the **Esc** key.

Depending on your model, you need to press this key during a long time. Also, your monitor can be too slow to be up while the message is on the screen. So, if you see your leds on your keyboard blinking, it is the signal to press and hold the **Esc** key. If you still have troubles, please refer to the *Consoles* section.

Now, there is some little differences about the way to get access to **BOOT_ADMIN** If you have an old box, you get a new Information message on your screen where the workstation−firmware told you that it would like to search for all bootable devices or you can break this by holding the **Esc** key. So you must do the same procedure you have done before, you must hold the **Esc** key.

Then you get a menu where you must push the a−key with following ENTER−Key, so know you are on the **BOOT_ADMIN** Prompt. First, we must turn off the auto booting by the following lines on the **BOOT_ADMIN** Prompt:

```
BOOT_ADMIN> autoboot off
```

then press the **ENTER** key.

After that, you must tell the system from wich SCSI boot device you would like to boot. This device must have the *f0* type partition where the **PALO** loader lives.

For this example, the old kernel is `vmlinux−2.4.9−32` and the new one for example `vmlinux−2.4.17−pa3`. The format of the SCSI boot device is: `SCSI.X.0` where *X* is the SCSI−ID of the disk you want. For example:

```
BOOT_ADMIN> boot SCSI.5.0
```

After the command **boot** and the SCSI−ID, you must add **IPL** if you have a HP 9000/7xx to tell that you want to interact with **IPL**. If you have a more recent hardware, the system will ask if you want to interact with **IPL**:

```
Interact with IPL (Y or N)?>
```

Now, you can manual configure the **PALO** booting parameters. For example:

```
BOOT_ADMIN> boot scsi.5.0 ipl
```

then press the **ENTER** key.

Now you can see a new Menu where you can configure on line 0 (default) the boot partition number, the path and the name where your boot kernel is. It should looks like this:

```
HARD Booted.
palo ipl 0.92 root@spqr Mon Jun 25 23:03:13 CEST 2001
3/vmlinux−2.4.9−32 3585851 bytes @ 0x6d8f800
Current command line:
3/vmlinux−2.4.9−32 root=/dev/sda3 console=ttyS0 TERM=vt102
 0: 3/vmlinux−2.4.9−32
 1: root=/dev/sda3
 2: console=ttyS0
 3: TERM=vt102

Edit which field?
(or 'b' to boot with this command line)? 0
```

You are asking **PALO** to boot the kernel file `vmlinux−2.4.9−32` living on the third partition of scsi.5.0. But you want another kernel this time. So, you press the **ENTER** key and you modify the text to match with your needs, here `vmlinux−2.4.17−pa3`. You validate your input *via* the **ENTER** key. Then, it asks you which field you want to edit, just put "b" to boot your new kernel! After that you must press the **ENTER** key, please don't chance other parameters if you don't know what you do ! Now, **PALO** has no secret for you :−)

---

6. What you need to know about BOOT_ADMIN

BOOT_ADMIN is an early console where you can execute some precise commands. Here, you should find everything you need. On all HP systems using PA−RISC processor, you will find **BOOT_ADMIN**, the display can be different but the idea remains. That's why the list is not complete but enough. Another major point is for every command, you have a shorter way to invocate them. You can see the shortcut because it is shown by the uppercase letters. I will put the full name for these sections.

6.1. The main commands

These commands are the basic ones.

- **boot** must be followed by an argument which indicates the path you want to boot. The path should be thedefinition of a device like for example `FWSCSI.6.0` or `PRI` if you have initialized this variable correctly.
- **path** displays or sets the previous path. You just have to type: `path pri fwscsi.6.0`. But, you can set also the paths of the console (graphics/serial) and of the keyboard (ps2/hil).
- **search** is a very useful command. It checks automatically and displays all the bootable pathes. In the recent versions, it links them to a shortcut. It detects even if the box can do **boot lan**.
- **display** redisplays the current menu.
- **help** gives you an overview of what the command does. By default, you can have all the commands by typing `help main`.
- On approximately every system, you have a **reset** instruction. It makes the box reboot with the newest parameters you hase set.

6.2. The configuration commands

These commands are available in the configuration menu. So, in order to use them, you must enter this menu with **configuration** command.

- **auto** will show you if the box will boot by default or will do a search. You can influence these parameters with the keywords *`ON`* and *`OFF`*.
- **bootinfo** lists all the boot parameters of the system.
- **default** sets back the predefined values.
- **monitor** sets your display configuration by typing `mo <path> <type>` which indicates your console path and secondly your type. If you do not know your monitor type, *via* `mo list`, you have all the necessary info.

6.3. The information commands

Here, you will have access to all the informations about your system. Go into the menu is done by asking for **information**.

- **all** should display everything you need.
- **fwrversion** gives your fimrware revision. You can check if your firmware is updated with this file.

* **lanaddress** contains the MAC address of the system.

6.4. The service commands

It is PA−RISC guru skill menu. You will find nothing really interesting for an end−user. That's why you will find nothing. If you think I should really put something, please mail me.

7. HOWTO contributors

The following people contributed or reviewed the Deb's version of the HOWTO in one way or another.

- David Alexander deVries <adevries@thepuffingroup.com>
- Philip Imperial Schwan <pschwan@thepuffingroup.com>

For the Thomas' version.

- Michael Damasche <sps01@uni-koeln.de> Thanks for your example
- Helge Deller <deller@gmx.de>
- Grant Grundler <grundler@puffin.external.hp.com>
- Richard Hirst <rhirst@puffin.external.hp.com>