

Troubleshooting boot problems on SUSE Linux Enterprise Server

Fixing a broken initrd

In most situations, your Linux server just keeps on running the way it should. In some situations, disaster strikes. After installing an updated driver, it has happened that the initrd was broken and the server rebooted into a "kernel panic" message. In this article you'll read what you can do in a situation like that.

Step 1: Determining what is wrong

The first step in troubleshooting the boot procedure is to determine what is wrong. This is a near-science by itself on which I could spend many articles, but let's try to say something about it in a generic way. During the system boot procedure, several phases occur. Roughly, these are the following:

1. GRUB loads the kernel
2. GRUB loads the initrd
3. The root file system is accessed by the kernel
4. The `/sbin/init` process takes over.
5. The initial boot stage happens
6. The default runlevel is activated
7. A login prompt occurs.

When a problem occurs, you need to pin-point it at any of these 7 phases. In some cases it is possible to tell exactly what happens, more often you will see that you can only give a rough indication of what is happening. In the case of a kernel panic, you can be sure about one thing: GRUB has loaded successfully and you are not yet at phase 4 of the boot procedure where the `init` process takes over. If a kernel panic occurs immediately after a driver installation, this is often caused by an error in the initrd. How to be sure about that?

Sometimes it is quite obvious that the error is in initrd, as GRUB tells you that it failed to load the file `/boot/initrd`, in other cases some more forensic work is needed as only a vague driver error message is generated. In the latter case, you have to check if the driver that fails is included in the initrd, as this helper file is used by the kernel to include drivers that are needed immediately. On SUSE Linux Enterprise Server, the file `/etc/sysconfig/kernel` contains a list of all drivers that should be included in the initrd. When you run the `mkinitrd` command, these drivers are written to your new initrd. When this happens automatically, something could go wrong.

Step 2: Fixing it

If an error occurs in the initrd, you will not be able to boot your server anymore. So, to fix it, you need the rescue system that is available from the installation dvd. This rescue system loads a Linux system that is completely on the installation media, so next you need to mount all your Linux file systems that are on disk. Next, you need to run `mkinitrd`. You can only do this once the local file systems are all mounted, because the initrd has to be written to the local file systems. There is however a caveat.

The problem with this approach is in the access to the disk devices in combination with the necessary use of a chroot environment. So let's see what's happening. To start, you need to mount your server's file systems on a temporary mount point like `/mnt`. Let's say that you have

the /boot directory on /dev/sda1 and your / directory on /dev/sda2. To mount them, you need the following two commands:

1. `mount /dev/sda2 /mnt`
2. `mount /dev/sda1 /mnt/boot`

Since the `mkinitrd` command wants to write the new `initrd` in /boot and the /boot on your hard drive is now in /mnt/boot, you now need to change the root directory to be set to /mnt. You can use `chroot` to do that:

```
chroot /mnt
```

The result of this command, is that the contents of /mnt now has become /, so all path references are OK. There is however still a problem. If you look in the /proc and /dev directory on your new root environment, you'll see that /proc is empty and /dev is as good as empty. Both are dynamically created file systems and they are created at the moment that your server boots. This means that they were created in / when the server booted from the rescue cd. Since now the new root is in /mnt, you can't access them anymore, so we need to fix this.

1. Type `exit` to exit from the `chroot` environment. You'll now get back to the original /mnt under which your servers local file systems were mounted.
2. Use `mount -t proc none /mnt/proc` to make the `proc` file system available from the /mnt environment.
3. Use `mount -o bind /dev /mnt/dev` which will make the original /dev which was filled by the `udev` process when booting available from /mnt/dev.

Now that you have the repair environment all in place, you need to check that the line in `/etc/sysconfig/kernel` that is used to generate a new `initrd` is as it should be. You are looking for the following line:

```
INITRD_MODULES="ata_piix processor thermal fan jbd ext3 dm_mod edd pciback"
```

This line will be different on every server, so check it well to make sure that all modules are included that you need to start your server (you do have your server's documentation to help you with that, don't you?)

Now under /mnt you have the complete environment that is needed to repair your server, so take the following two steps to fix your server.

1. Activate /mnt using `cd /mnt` and make it your new root environment using `chroot`.
2. Issue the command `mkinitrd` to write the new `initrd` to /boot.

You have now fixed the `initrd`. Reboot your server and check that everything is working all right.

Summary

Troubleshooting a server can be cumbersome as many problems with a different character can occur. In this article you have learned how to fix your server when the `initrd` fails. This

information helps you in applying a quick fix to your server when something went wrong during an upgrade, thus avoiding to loose a lot of money because of unexpected down time.

Sander van Vugt