When attempting to boot your Linux system, you see an error message and the system will not complete a normal boot. Which of the following could you type at the GRUB menu to enter a system admin mod for troubleshooting:

1 (s)(S)

The numeral 1 puts the system into runlevel 1, a troubleshooting mode. The character s or S also are used to put the system into a troubleshooting mode.

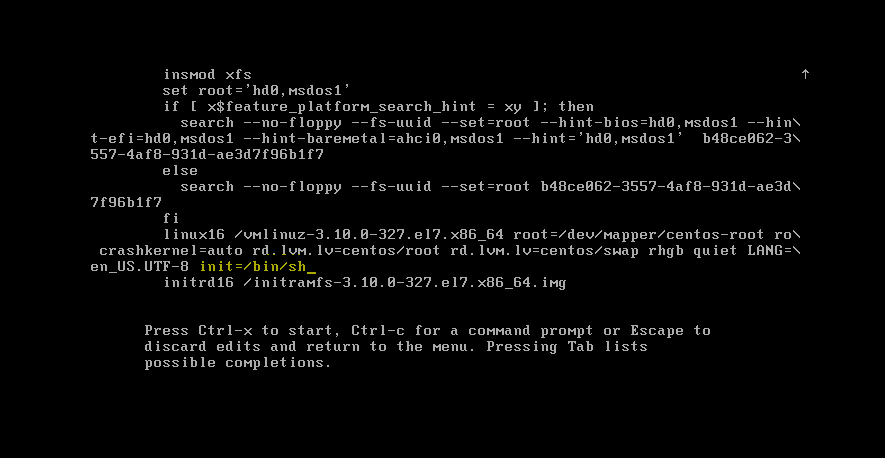
If the above method does not work, you can use below method

The system is having trouble and the engineer wants to bypass the usual /sbin/init start up and run /bin/sh. What is the usual way to pass this change to the kernel from your boot loader:

Pass init=/bin/sh on the kernel parameter line

Step 1: While standing on Grub’s menu, press e to edit

Step 2: Pass init=/bin/sh on the kernel parameter line



In case you use above method, you have to mount -o remount,rw / to write /etc/fstab

i did it like this..  
1.boot with a instalation CD and go on to 'linux rescue' mode  
2. mount the root partition to some other path other than root (as it is already mounted)  
say mount on to /mnt/root  
3. and then i got the /etc/fstab editable.

**Recovery**

Now, let's look at some things that can go wrong with your carefully prepared boot setup, particularly when you install and boot multiple operating systems. The first thing to remember is to resist your initial temptation to panic. Recovery is usually only a few steps away. The strategies here can help you through many types of crises.

Anyone with physical access to a machine has a lot of power. Likewise, anyone with access to a GRUB command line also has access to files on your system without the benefit of any ownership or other security provisions provided by a running system. Keep these points in mind when you select your boot loader. The choice between LILO, GRUB, or GRUB 2 is largely a matter of personal preference, although GRUB 2 is becoming dominant. Choose the loader that best suits your particular needs and style of working.

**Another install destroys your MBR**

You install another operating system and inadvertently overwrite your MBR. Some systems, such as DOS and Windows, always install their own MBR. It is usually easy to recover from this situation. If you develop a habit of creating a recovery floppy, USB flash drive, or CD every time you run lilo, reinstall GRUB, or update GRUB 2, you are home free. Boot into your Linux system from the floppy and rerun lilo, grub-install or grub2-install as appropriate.

If you don't happen to have your own recovery media but you still have almost any Linux distribution live or install media available, you can either use the recovery mode of the distribution media or the live media to either repair your broken MBR or to build recovery media as you have done in this tutorial.

**You moved a partition**

If you moved a partition and forgot about your boot setup, you have a temporary problem. Typically, LILO or GRUB refuse to load. LILO will probably print an 'L' indicating that stage 1 was loaded and then stop. GRUB will give you an error message. What has happened here is that the stage 1 loader, which had a list of sectors to load to get to the stage 2 loader, can perhaps load the sectors from the addresses it has, but the sectors no longer have the stage 2 signature. As for the case of the destroyed MBR, you need to reinstall your boot loader, so either use your recovery CD or a Linux distribution as I described.

You might have noticed that the configuration examples used some Universally Unique IDs (UUIDs) for partitions (for example, the snippet shown in [Listing 17](http://www.ibm.com/developerworks/library/l-lpic1-102-2/#using-uuid) that you saw in [Listing 1](http://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub-menu-lst).

**Listing 17. Using a UUID**

title CentOS (2.6.32-504.23.4.el6.x86\_64)

root (hd0,10)

kernel /boot/vmlinuz-2.6.32-504.23.4.el6.x86\_64 ro \

root=UUID=2f60a3b4-ef6c-4d4c-9ef4-50d7f75124a2 rd\_NO\_LUKS rd\_NO\_LVM \

LANG=en\_US.UTF-8 rd\_NO\_MD SYSFONT=latarcyrheb-sun16 crashkernel=128M \

KEYBOARDTYPE=pc KEYTABLE=us rd\_NO\_DM rhgb quiet

initrd /boot/initramfs-2.6.32-504.23.4.el6.x86\_64.img

Prior to the advent of UUIDs, MBR partitions could also have a label assigned using the e2label command or a partitioning tool such as gparted. These also provide a level of independence from partition moves.

I often use UUIDs like this to help avoid problems when I move partitions. You still need to update the GRUB or LILO configuration file and rerun lilo, but you don't have to update /etc/fstab as well. This is particularly handy if you create a partition image on one system and restore it at a different location on another system. It's also handy if you boot from a drive, such as a USB drive, that might not always be attached at the same location.

You can use the blkid command to display the labels (if any) and UUIDs for your disks as shown in [Listing 18](http://www.ibm.com/developerworks/library/l-lpic1-102-2/#use-blkid). As you can see, some of my partitions use labels and some don't.

**Listing 18. Displaying labels and UUIDs for partitions**

ian@attic-u14:~$ **sudo blkid**

/dev/sda1: LABEL="/grubfile" UUID="3c3de27e-779a-44d5-ad7a-61c5fd03d9e7" TYPE="ext3"

/dev/sda2: UUID="158d605e-2591-4749-bf59-5e92e1b1c01d" TYPE="swap"

/dev/sda3: UUID="ff0b87d2-6929-45df-88e1-d6d3e5cf3d6f" TYPE="ext4"

/dev/sda5: LABEL="FEDORA22" UUID="7aefe7a0-97d5-45ec-a92e-00a6363fb1e4" TYPE="ext4"

/dev/sda6: UUID="78a8c7de-cb86-45fe-ac04-be67ef52cb12" TYPE="ext4"

/dev/sda7: LABEL="FEDORA 18" UUID="1b441a69-63e3-4771-a06b-5efecd1df07e" TYPE="ext4"

/dev/sda8: LABEL="SUSE13-2" UUID="49d87897-791e-4e48-9efb-704eac447e43" SEC\_TYPE="ext2" TYPE="ext3"

/dev/sda9: UUID="10e82894-186f-4223-95c8-3468eb9b085d" SEC\_TYPE="ext2" TYPE="ext3"

/dev/sda10: LABEL="FEDORA20-64" UUID="8e6e2ebd-20b9-46e8-865f-893dd88c3206" TYPE="ext4"

/dev/sda11: UUID="2f60a3b4-ef6c-4d4c-9ef4-50d7f75124a2" SEC\_TYPE="ext2" TYPE="ext3"

/dev/sda12: LABEL="UBUNTU-1404" UUID="943524cc-19a9-4237-ac9e-5c1a61a131e3" TYPE="ext4"

/dev/sdb1: LABEL="GRUB-DATA" UUID="a36a3539-8393-4940-a893-472e9e1c868e" SEC\_TYPE="ext2" TYPE="ext3"

/dev/sdb2: LABEL="DATA-IAN" UUID="4c962b67-c646-467f-96fb-cbbd6de40140" TYPE="ext4"

/dev/sdb3: LABEL="RESEARCH" UUID="0998d33c-3398-463d-b0e3-7c13ca0c675f" TYPE="ext3"

/dev/sdb4: UUID="86ad1df3-fea4-47e5-bfdd-fb09f6c2e64a" TYPE="ext4"

/dev/sdc1: LABEL="PICTURES" UUID="e3be4658-b79b-470d-82fe-bb434bcdcc2f" TYPE="ext4"

/dev/sr0: LABEL="ISOIMAGE" TYPE="iso9660"

GRUB 2 nowadays generates configuration files that use the old device names as hints, but the root is actually set based on the UUID (or label). Consult the GRUB 2 manual for more details. An example snippet for the Ubuntu 14 system that I built a configuration for earlier is shown in [Listing 19](http://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub2-search-uuid). After initially setting the root to 'hd0,msdos12', the logic following uses the search command with some hints to locate the actual location of the root device (/dev/sda12 if nothing has changed).

**Listing 19. Locating the root device by UUID using GRUB2**

set root='hd0,msdos12'

if [ x$feature\_platform\_search\_hint = xy ]; then

search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos12 --hint-efi=hd0,msdos12 \

--hint-baremetal=ahci0,msdos12 943524cc-19a9-4237-ac9e-5c1a61a131e3

else

search --no-floppy --fs-uuid --set=root 943524cc-19a9-4237-ac9e-5c1a61a131e3

fi

You saw this in use when you booted from the recovery flash drive earlier. Remember that the flash drive became (hd0) and the first hard drive became (hd1).

In addition to the methods and tools mentioned here, there are several recovery and boot packages available on the Internet. These typically include some level of bootable system along with a number of useful recovery tools. Examples include packages like Knoppix and the System Rescue CD (see [Resources](http://www.ibm.com/developerworks/library/l-lpic1-102-2/#resources) for links, or search the Internet for one of many excellent reviews of such packages.

**Using a boot partition**

Another approach to recovery, or perhaps avoiding it, is to use a separate partition for booting. As you just saw, GRUB 2 has become much more resilient to system changes. However for GRUB Legacy and LILO, a boot partition can be particularly useful if you have a test system with several distributions on it that you might rebuild frequently. The boot partition need not be very large, perhaps 100MB or so. Put this partition somewhere where it is unlikely to be moved and where it is unlikely to have its partition number moved by the addition or removal of another partition. In a mixed Windows and Linux environment, /dev/sda2 (or /dev/hda2 depending on how your disks are labeled) is often a good choice for your boot partition. In fact, [Listing 1](http://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub-menu-lst) that you saw earlier shows the entries in the small boot partition (/dev/sda1) that I use on my system.

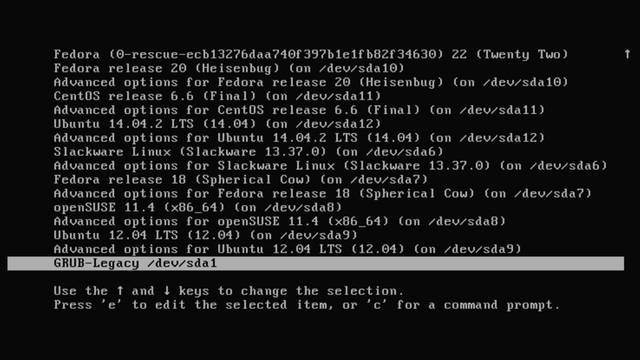
Although I added an entry for the purpose of this tutorial to directly boot into CentOS, my usual strategy is to keep the entries simple and to use them to chain load another boot loader, or load the GRUB 2 core.img file. The examples that you saw in the section [Booting GRUB 2 from Grub Legacy and vice-versa](http://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub-to-grub2-and-back) use this simple strategy. If you do use entries that directly boot into a particular kernel you need to manually update them whenever you update the target system. Avoid the extra work when you can. [Figure 8](http://www.ibm.com/developerworks/library/l-lpic1-102-2/#boot-part-grub) shows my simple GRUB menu.

**Figure 8. A simple GRUB boot partition menu**



For this partition, I use a custom splash image that I made from a photo I took in the Great Sand Dunes National Park in Colorado. Consult the man pages or search online for more information on making your own splash images. Figure 9 shows the entry that was added to the end of my Fedora 22 GRUB 2 menu using the /etc/grub.d/40\_custom file.

**Figure 9. Chain loader entry from GRUB from GRUB 2**



Another reason for having a boot partition arises when your root partition uses a file system not supported by your boot loader. For example, it is common to have a /boot partition formatted ext2 or ext3 when the root partition (/) uses LVM, which is not supported by GRUB Legacy.

If you have multiple distributions on your system, **do not** share the /boot partition between them. Remember to set up LILO or GRUB to boot from the partition that will later be mounted as /boot. Remember also that the update programs for a distribution usually update the GRUB or LILO configuration for that system. In an environment with multiple systems, you might want to keep one with its own /boot partition as the main one and manually update that configuration file whenever an update of one of your systems requires it. Another approach is to have each system install a boot loader into its own partition boot record and have your main system simply chain load the partition boot records for the individual systems, giving you a two-stage menu process like the one I use.