1. About the Install

First, if you are new to this, welcome to Gentoo Linux! Gentoo Linux can be installed in many different ways. Those who are looking for a rapid install can use pre-built packages, while those who want the ultimate in customizability can compile Gentoo Linux entirely from the original source code. The method you choose is up to you.

One significant change in relation to the official 1.4 release is our new 2-CD installation set, which can be ordered from The Gentoo Linux Store, in addition to being available on our mirrors. We currently have 2-CD installation sets for x86 (486 and above), i686 (Pentium Pro, Pentium II, Athlon/Duron and above), Pentium III, Pentium 4 and Athlon XP. To see what 2-CD set is right for you, read the detailed descriptions of each product in the store. The store descriptions contain fairly comprehensive CPU compatibility information.

So, about the 2 CD set -- here's what's on each CD. The first CD ("CD 1") is called "Live CD Installation" and is a bootable CD-ROM, meaning that you can put "CD 1" in your drive and run Gentoo Linux directly from the CD. You can then use this CD-based version of Gentoo to install Gentoo Linux 1.4 to your hard disk. In addition to containing a bootable Gentoo Linux environment, CD 1 contains everything you need to install Gentoo Linux quickly, even without a connection to the Internet. In addition, several pre-compiled packages are also included on CD 1, such as the ever-important XFree86 X server. If you have an ISO CD-ROM image file for CD 1, its name will end in "`-cd1.iso`".

In contrast, the second CD ("CD 2") isn't bootable and contains lots of pre-built packages for your system. Included on this CD are optimized versions of packages such as KDE, GNOME, OpenOffice, Mozilla, Evolution and others. CD 2 is **optional** and is intended for those people who are interested in installing Gentoo Linux very quickly. The packages included on CD 2 typically take about 36 hours to compile from source on a typical modern single-processor system. If you have an ISO CD-ROM image file for CD 2, its name will end in "`-cd2.iso`".

Note: A complete Gentoo Linux 2-CD set contains the Gentoo Reference Platform, which is a complete pre-built Gentoo Linux system including GNOME, KDE, Mozilla and OpenOffice. The Gentoo Reference Platform ("GRP") was created to allow rapid Gentoo Linux package installations for those who need this capability. The "compile from source" functionality, which is the cornerstone of Gentoo Linux, will always be a fully-supported installation option as well. The purpose of the GRP is to make Gentoo Linux more convenient for some users, without impacting Gentoo's powerful "compile from source" installation process in any way.

In addition to our 2-CD set, we also have a very small "basic" Live CD that you can use to boot your system. Once your system has booted, you can configure a connection to the Internet and then install Gentoo over the network. The advantage of this "basic" CD is that it is small and thus the ISO CD-ROM image file can be downloaded quickly. If you're an advanced user who wants to install the most up-to-date version of Gentoo Linux available and have a fast network connection, then you may prefer this option. If you have an ISO CD-ROM image file for our "basic" Live CD, its name will end in "`-basic.iso`".

To use any Gentoo Linux CD-based installation method, you will need to have a 486+ processor and ideally at least 64 Megabytes of RAM. (Gentoo Linux has been successfully built with 64MB of RAM + 64MB of swap space, but the build process is awfully slow under those conditions.)

Once you boot one of our Live CDs, you have even more options. Gentoo Linux can be installed using one of three "stage" tarball files. The one you choose depends on how much of the system you want to compile yourself. The stage1 tarball is used when you want to bootstrap and build the entire system from scratch. The stage2 tarball is used for building the entire system from a bootstrapped "semi-compiled" state. The stage3 tarball already contains a basic Gentoo Linux system that has been built for you. If you are interested in doing a "GRP" install, then the stage3 tarball should be used.

If you're not doing a GRP install, should you start from a stage1, stage2, or stage3 tarball? Here is some information that should help you make this decision.

Starting from a stage1 allows you to have total control over the optimization settings and optional build-time functionality that is initially enabled on your system. This makes stage1 installs good for power users who know what they are doing. It is also a great installation method for those who would like to know more about the inner workings of Gentoo Linux.

Stage2 installs allow you to skip the bootstrap process and doing this is fine if you are happy with the optimization settings that we chose for your particular stage2 tarball.

And choosing to go with a stage3 allows for the fastest install of Gentoo Linux, but also means that your base system will have the optimization settings that we chose for you (which to be honest, are good settings and were carefully chosen to enhance performance while maintaining stability). Since major releases of Gentoo Linux have stage3's specifically optimized for various popular processors, starting from a stage3 can offer the best of all worlds -- a fast install and a system that is well-optimized.

If you're installing Gentoo Linux for the first time, consider using a stage3 tarball for installation, or a stage3 with GRP.

Note: Advanced users: If you use a stage3 install, you should not change the default CHOST setting in `/etc/make.conf`. If you need to make such a change, you should start with a stage1 tarball and build up your system with the desired CHOST setting. The CHOST setting typically looks something like this: **`i686-pc-linux-gnu`**.

Note: If you encounter a problem with any part of the install and wish to report it as a bug, report it to http://bugs.gentoo.org. If the bug needs to be sent upstream to the original software developers (e.g. the KDE team) the **Gentoo Linux developers** will take care of that for you.

Note: The installation instructions in the LiveCD may not be as up-to-date as our Web documentation at http://www.gentoo.org/doc/en/gentoo-x86-install.xml. Refer to our Web documentation for the most up-to-date installation instructions.

Now, let us quickly review the install process. First, we will download, burn CD(s) and boot a LiveCD. After getting a root prompt, we will create partitions, create our filesystems and extract either a stage1, stage2 or stage3 tarball. If we are using a stage1 or stage2 tarball, we will take the appropriate steps to get our system to stage3. Once our system is at stage3, we can configure it (customize configuration files, install a boot loader, etc.), boot it and have a fully-functional Gentoo Linux system. After your basic Gentoo Linux system is running, you can optionally use "CD 2" of our 2-CD set and install any number of pre-built packages such as KDE, GNOME, OpenOffice, Mozilla, or others that you'd like on your system.

Depending on what stage of the build process you're starting from, here is what is required for installation:

Stage Tarball	Internet Access Required	Media Required	Steps
1	Yes	**basic** or **CD 1**	partition/filesystem setup, emerge sync, bootstrap, emerge system, final config
2	Yes	**basic** or **CD 1**	partition/filesystem setup, emerge sync, emerge system, final config

3	No if using **CD 1**, Yes otherwise	**basic** or **CD 1**	partition/filesystem setup, emerge sync (not required if using **CD 1**), final config
3+GRP	No	**CD 1**, **CD 2** optionally	partition/filesystem setup, final config, install CD 1 pre-built packages (optional), reboot, install extra pre-built packages like KDE and GNOME (if using "CD 2")

Note: Hardware ATA RAID users should read the section about ATA RAID on the bottom of this document before proceeding.

2. Booting

Note: Read this whole section before proceeding, especially the available boot options. Ignoring this could lead to wrong keyboard settings, unstarted pcmcia services etc..

Start by booting your Live CD of choice. You should see a fancy boot screen with the Gentoo Linux logo on it. At this screen, you can hit Enter to begin the boot process, or boot the LiveCD with custom boot options by specifying a kernel followed by boot options and then hitting Enter. For example: **`gentoo nousb nohotplug`**. If you are installing Gentoo Linux on a system with more than one processor ("SMP"), then you should type **`smp`** instead of **`gentoo`** at the prompt. This will allow the LiveCD to see all the processors in your system, not just the first one.

Consult the following table for a partial list of available kernels and options or press F2 and F3 to view the help screens.

Available kernels	Description
gentoo	standard gentoo kernel (default)
nofb	framebuffer mode disabled
smp	loads a smp kernel in noframebuffer mode
acpi	enables acpi=on + loads acpi modules during init
memtest	boots the memory testing program

Available boot options	Description
doataraid	loads ide raid modules from initrd
dofirewire	modprobes firewire modules in initrd (for firewire cdroms,etc.)
dokeymap	enable keymap selection for non-us keyboard layouts
dopcmcia	starts pcmcia service
doscsi	scan for scsi devices (breaks some ethernet cards)
noapm	disables apm module load
nodetect	causes hwsetup/kudzu and hotplug not to run
nodhcp	dhcp does not automatically start if nic detected
nohotplug	disables loading hotplug service
noraid	disables loading of evms modules
nousb	disables usb module load from initrd, disables hotplug
ide=nodma	force disabling of dma for malfunctioning ide devices
cdcache	cache the entire runtime portion of cd in ram. This uses 40mb of RAM, but allows you to umount `/mnt/cdrom` and mount another cdrom

Once you hit Enter, you will be greeted with an even fancier boot screen and progress bar.

Once the boot process completes, you will be automatically logged in to the "Live" Gentoo Linux as "**root**", the "super user". You should have a root ("#") prompt on the current console and can also switch to other consoles by pressing Alt-F2, Alt-F3 and Alt-F4. Get back to the one you started on by pressing Alt-F1.

Note: Advanced users: When the Live CD boots, the Live CD root password is set to a random string for security purposes. If you plan to start **`sshd`** to allow remote logins to your Live CD, you should set the Live CD root password now by typing **`passwd`** and following the prompts. Otherwise, you will not know the proper password for logging into the Live CD over the network.

You've probably also noticed that above your # prompt is a bunch of help text that explains how to do things like configure your Linux networking and telling you where you can find the Gentoo Linux stage tarballs and packages on your CD.

3. Optional hardware configuration

When the Live CD boots, it tries to detect all your hardware devices and loads the appropiate kernel modules to support your hardware. In the vast majority of cases, it does a very good job. However, in some cases, it may not auto-load the kernel modules you need. If the PCI auto-detection missed some of your system's hardware, you will have to load the appropriate kernel modules manually. To view a list of all available network card modules, type **`ls /lib/modules/`uname -r`/kernel/drivers/net/*`**. To load a particular module, type:

Code listing: PCI Modules Configuration

```
# modprobe pcnet32
(replace pcnet32 with your NIC module)
```

Likewise, if you want to be able to access any SCSI hardware that wasn't detected during the initial boot autodetection process, you will need to load the appropriate modules from `/lib/modules`, again using **`modprobe`**:

Code listing: Loading SCSI Modules

```
# modprobe aic7xxx
(replace aic7xxx with your SCSI adapter module)
# modprobe sd_mod
(sd_mod is the module for SCSI disk support)
```

Note: Support for SCSI CD-ROMs and disks are built-in in the kernel.

Note: Advanced users: The Gentoo LiveCD should have enabled DMA on your disks so that disk transfers are as fast as possible, but if it did not, **hdparm** can be used to set DMA on your drives as follows:

Code listing: Setting DMA

```
(Replace hdX with your disk device)
# hdparm -d 1 /dev/hdX
(Enables DMA)
# hdparm -d1 -A1 -m16 -u1 -a64 /dev/hdX
(Enables DMA and other safe performance-enhancing options)
# hdparm -X66 /dev/hdX
(Force-enables Ultra-DMA -- dangerous -- may cause some drives to mess up)
```

4. Optional Networking configuration

Maybe it just works?

If your system is plugged into an Ethernet network, it is very likely that your networking configuration has already been set up automatically for you. If so, you should be able to take advantage of the many included network-aware commands on the LiveCD such as **ssh**, **scp**, **ping**, **irssi**, **wget** and **lynx**, among others.

If networking has been configured for you, the **/sbin/ifconfig** command should list some internet interfaces besides lo, such as eth0:

Code listing: /sbin/ifconfig for a working network card

```
eth0      Link encap:Ethernet  HWaddr 00:50:BA:8F:61:7A
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::50:ba8f:617a/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1498792 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1284980 errors:0 dropped:0 overruns:0 carrier:0
          collisions:1984 txqueuelen:100
          RX bytes:485691215 (463.1 Mb)  TX bytes:123951388 (118.2 Mb)
          Interrupt:11 Base address:0xe800
```

You may want to also try pinging your ISP's DNS server (found in `/etc/resolv.conf`) and a Web site of choice, just to make sure that your packets are reaching the net, DNS name resolution is working correctly, etc..

Code listing: Further Network Testing

```
# ping -c 3 www.yahoo.com
```

Are you able to use your network? If so, you can skip the rest of this section.

PPPoE configuration

Assuming you need PPPoE to connect to the internet, the LiveCD (any version) has made things easy for you by including **rp-pppoe**. Use the provided **adsl-setup** script to configure your connection. You will be prompted for the ethernet device that is connected to your adsl modem, your username and password, the IPs of your DNS servers and if you need a basic firewall or not.

Code listing: Configuring PPPoE

```
#  adsl-setup
#  adsl-start
```

If something goes wrong, double-check that you correctly typed your username and password by looking at `/etc/ppp/pap-secrets` or `/etc/ppp/chap-secrets` and make sure you are using the right ethernet device.

Automatic Network Configuration

The simplest way to set up networking if it didn't get configured automatically is to run the **net-setup** script:

Code listing: Net-Setup Script

```
# net-setup eth0
```

Of course, if you prefer, you may still set up networking manually. This is covered next.

Manual DHCP Configuration

Network configuration is simple with DHCP; If your ISP is not using DHCP, skip down to the static configuration section below.

Code listing: Network configuration with DHCP

```
# dhcpcd eth0
```

Note: Some ISPs require you to provide a hostname. To do that, add a **-h myhostname** flag to the dhcpcd command line above.

If you receive **dhcpConfig** warnings, don't panic; the errors are most likely cosmetic. Skip down to Network testing below.

Manual Static Configuration

We need to setup just enough networking so that we can download sources for the system build, as well as the required localhost interface. The needed information is explained in the next table.

Information	Description	Example value
IP address	The IP address you want to assign to your network card	192.168.1.2
Broadcast address	The IP address which will broadcast the packets to all the hosts in the network	192.168.1.255
Network mask	The mask which is used together with the IP address to see what part of the address is for network-identification and host-identification	255.255.255.0
Gateway	The IP address of the computer which will forward the packets that are not meant for the local network (most of the time the computer which shares the internet connection)	192.168.1.1

Type in the following commands, replacing **$IFACE** with your network interface (typically **eth0**), **$IPNUM** with your IP address, **$BCAST** with your broadcast address and **$NMASK** with your network mask. For the **route** command, replace **$GTWAY** with your default gateway.

Code listing: Static IP Network Configuration

```
# ifconfig $IFACE $IPNUM broadcast $BCAST netmask $NMASK
# route add -net default gw $GTWAY netmask 0.0.0.0 metric 1 $IFACE
```

Now it is time to create the `/etc/resolv.conf` file so that name resolution (finding Web/FTP sites by name, rather than just by IP address) will work. You can use **nano -w /etc/resolv.conf** to create `/etc/resolv.conf`. **nano** is a small and easy-to-use editor.

Here is a template to follow for creating your `/etc/resolv.conf` file:

Code listing: /etc/resolv.conf template

```
domain mydomain.com
nameserver 10.0.0.1
nameserver 10.0.0.2
```

Replace **10.0.0.1** and **10.0.0.2** with the IP addresses of your primary and secondary DNS servers respectively.

Proxy Configuration

If you are behind a proxy, it could be necessary to configure your proxy before you continue. We will export some variables to set up the proxy accordingly.

Code listing: Setting a Proxy

```
If the proxy restricts HTTP traffic:
# export http_proxy="http://machine.company.com:1234"
If the proxy restricts FTP traffic:
# export ftp_proxy="ftp://machine.company.com"
If the proxy restricts RSYNC traffic:
# export RSYNC_PROXY="rsync://machine.company.com"
```

Networking is go!

Networking should now be configured and usable. You should be able to use the included **ssh**, **scp**, **lynx**, **irssi** and **wget** commands to connect to other machines on your LAN or the Internet.

5. Setting your system's date and time

Now you need to set your system's date and time. You can do this using the **date** command.

Code listing: Setting your system's date

```
# date
Thu Feb 27 09:04:42 CST 2003
(If your date is wrong, set your date with this next command)
# date 022709042003
(date MMDDhhmmCCYY)
```

6. Filesystems, partitions and block devices

Introduction to block devices

In this section, we'll take a good look at disk-oriented aspects of Gentoo Linux and Linux in general, including Linux filesystems, partitions and block devices. Then, once you're familiar with the ins and outs of disks and filesystems, you'll be guided through the process of setting up partitions and filesystems for your Gentoo Linux installation.

To begin, I'll introduce "block devices". The most famous block device is probably the one that represents the first IDE drive in a Linux system:

Code listing: /dev/hda, the block device representing the primary master IDE drive in your system

```
/dev/hda
```

If your system uses SCSI drives, then your first hard drive will be:

Code listing: /dev/sda, the block device representing the first logical SCSI drive in your system

```
/dev/sda
```

The block devices above represent an **abstract** interface to the disk. User programs can use these block devices to interact with your disk without worrying about whether your drives are IDE, SCSI or something else. The program can simply address the storage on the disk as a bunch of contiguous, randomly-accessible 512-byte blocks.

Partitions and fdisk

Under Linux, we create filesystems by using a special command called **mkfs** (or **mke2fs**, **mkreiserfs**, etc.), specifying a particular block device as a command-line argument.

However, although it is theoretically possible to use a "whole disk" block device (one that represents the **entire** disk) like `/dev/hda` or `/dev/sda` to house a single filesystem, this is almost never done in practice. Instead, full disk block devices are split up into smaller, more manageable block devices called "partitions". Partitions are created using a tool called **fdisk**, which is used to create and edit the partition table that's stored on each disk. The partition table defines exactly how to split up the full disk.

We can take a look at a disk's partition table by running **fdisk**, specifying a block device that represents a full disk as an argument:

Note: Alternate interfaces to the disk's partition table include **cfdisk**, **parted** and **partimage**. We recommend **fdisk** because it's more powerful and well known in the Unix/Linux world.

Code listing: Starting up fdisk

```
# fdisk /dev/hda
```

or

Code listing: Starting up fdisk to look at the partition table on /dev/sda

```
# fdisk /dev/sda
```

Note: Note that you should **not** save or make any changes to a disk's partition table if any of its partitions contain filesystems that are in use or contain important data. Doing so will generally cause data on the disk to be lost.

Once in **fdisk**, you'll be greeted with a prompt that looks like this:

Code listing: The fdisk prompt

```
Command (m for help):
```

Type **p** to display your disk's current partition configuration:

Code listing: An example partition configuration

```
Command (m for help): p

Disk /dev/hda: 240 heads, 63 sectors, 2184 cylinders
Units = cylinders of 15120 * 512 bytes

Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1        14    105808+  83  Linux
/dev/hda2            15        49    264600   82  Linux swap
/dev/hda3            50        70    158760   83  Linux
/dev/hda4            71      2184  15981840    5  Extended
/dev/hda5            71       209   1050808+  83  Linux
/dev/hda6           210       348   1050808+  83  Linux
/dev/hda7           349       626   2101648+  83  Linux
/dev/hda8           627       904   2101648+  83  Linux
/dev/hda9           905      2184   9676768+  83  Linux

Command (m for help):
```

This particular disk is configured to house seven Linux filesystems (each with a corresponding partition listed as "Linux") as well as a swap partition (listed as "Linux swap").

Notice the name of the corresponding partition block devices on the left hand side, starting with `/dev/hda1` and going up to `/dev/hda9`. In the early days of the PC, partitioning software only allowed a maximum of four partitions (called "primary" partitions). This was too limiting, so a workaround called **extended partitioning** was created. An extended partition is very similar to a primary partition and counts towards the primary partition limit of four. However, extended partitions can hold any number of so-called **logical** partitions inside them, providing an effective means of working around the four partition limit.

All partitions `/dev/hda5` and higher are logical partitions. The numbers 1 through 4 are reserved for primary or extended partitions.

So, In our example, `/dev/hda1` through `/dev/hda3` are primary partitions. `/dev/hda4` is an extended partition that contains logical partitions `/dev/hda5` through `/dev/hda9`. You would never actually **use** `/dev/hda4` for storing any filesystems directly -- it simply acts as a container for partitions `/dev/hda5` through `/dev/hda9`.

Also, notice that each partition has an "Id", also called a "partition type". Whenever you create a new partition, you should ensure that the partition type is set

correctly. '83' is the correct partition type for partitions that will be housing Linux filesystems, '82' is the correct partition type for Linux swap partitions and 'fd' is the recommended partition type for Software RAID partitions. You set the partition type using the **t** option in **fdisk**. The Linux kernel uses the partition type setting to auto-detect filesystems and swap devices on the disk at boot-time.

Using fdisk to set up partitions

Now that you've had your introduction to the way disk partitioning is done under Linux, it's time to walk you through the process of setting up disk partitions for your Gentoo Linux installation. After we walk you through the process of creating partitions on your disk, your partition configuration will look like this:

Code listing: The partition configuration that you will have after following these steps

```
Disk /dev/hda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1        14    105808+  83  Linux
/dev/hda2            15        81    506520   82  Linux swap
/dev/hda3            82      3876  28690200   83  Linux

Command (m for help):
```

In our suggested "newbie" partition configuration, we have three partitions. The first one (`/dev/hda1`) at the beginning of the disk is a small partition called a boot partition. The boot partition's purpose is to hold all the critical data related to booting -- GRUB boot loader information (if you will be using GRUB) as well as your Linux kernel(s). The boot partition gives us a safe place to store everything related to booting Linux. During normal day-to-day Gentoo Linux use, your boot partition should remain **unmounted** for safety. If you are setting up a SCSI system, your boot partition will likely end up being `/dev/sda1`.

It's recommended to have boot partitions (containing everything necessary for the boot loader to work) at the beginning of the disk. While not necessarily required anymore, it is a useful tradition from the days when the lilo boot loader wasn't able to load kernels from filesystems that extended beyond disk cylinder 1024.

The second partition (`/dev/hda2`) is used to for swap space. The kernel uses swap space as virtual memory when RAM becomes low. This partition, relatively speaking, isn't very big either, typically somewhere around 512MB. If you're setting up a SCSI system, this partition will likely end up being called `/dev/sda2`.

The third partition (`/dev/hda3`) is quite large and takes up the rest of the disk. This partition is called our "root" partition and will be used to store your main filesystem that houses Gentoo Linux itself. On a SCSI system, this partition would likely end up being `/dev/sda3`.

Before we partition the disk, here's a quick technical overview of the suggested partition and filesystem configuration to use when installing Gentoo Linux:

Partition	Size	Type	example device
boot partition, containing kernel(s) and boot information	32 Megabytes	ext2/3 highly recommended (easiest); if ReiserFS then mount with **-o notail**. If you will be using ext3 or ReiserFS, you must add the size of the journal to the partitionsize; in these cases 64 Megabytes is recommended	`/dev/hda1`
swap partition (no longer a 128 Megabyte limit, now 2GB)	Generally, configure a swap area that is between one and two times the size of the physical RAM in your system	Linux swap	`/dev/hda2`
root partition, containing main filesystem (/usr, /home, etc.)	>=1.5 Gigabytes	ReiserFS, ext3 recommended; ext2 ok	`/dev/hda3`

OK, now to create the partitions as in the example and table above. First, enter fdisk by typing **fdisk /dev/hda** or **fdisk /dev/sda**, depending on whether you're using IDE or SCSI. Then, type **p** to view your current partition configuration. Is there anything on the disk that you need to keep? If so, stop now. If you continue with these directions, all existing data on your disk will be erased.

Note: Following these instructions below will cause all prior data on your disk to be erased! If there is anything on your drive, please be sure that it is non-critical information that you don't mind losing. Also make sure that you have selected the correct drive so that you don't mistakenly wipe data from the wrong drive.

Now, it's time to delete any existing partitions. To do this, type **d** and hit Enter. You will then be prompted for the partition number you would like to delete. To delete a pre-existing `/dev/hda1`, you would type:

Code listing: Deleting a partition

```
Command (m for help): d
Partition number (1-4): 1
```

The partition has been scheduled for deletion. It will no longer show up if you type **p**, but it will not be erased until your changes have been saved. If you made a mistake and want to abort without saving your changes, type **q** immediately and hit enter and your partition will not be deleted.

Now, assuming that you do indeed want to wipe out all the partitions on your system, repeatedly type **p** to print out a partition listing and then type **d** and the number of the partition to delete it. Eventually, you'll end up with a partition table with nothing in it:

Code listing: An empty partition table

```
Disk /dev/hda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes
```

```
Device Boot    Start       End    Blocks   Id  System

Command (m for help):
```

Now that the in-memory partition table is empty, we're ready to create a boot partition. To do this, type **n** to create a new partition, then **p** to tell fdisk you want a primary partition. Then type **1** to create the first primary partition. When prompted for the first cylinder, hit enter. When prompted for the last cylinder, type **+32M** to create a partition 32MB in size. You can see output from these steps below:

Note: Journaled filesystems require extra space for their journal. Default settings require about 33 Megabytes of space. Therefore, if you are using a journaled filesystem for `/boot`, you should type **+64M** when prompted for the last cylinder.

Code listing: Steps to create our boot partition

```
Command (m for help): n
Command action
e   extended
p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-3876, default 1): (Hit Enter)
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-3876, default 3876): +32M
```

Now, when you type **p**, you should see the following partition printout:

Code listing: Our first partition has been created

```
Command (m for help): p

Disk /dev/hda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot    Start       End    Blocks   Id  System
/dev/hda1          1        14    105808+  83  Linux
```

Next, let's create the swap partition. To do this, type **n** to create a new partition, then **p** to tell fdisk that you want a primary partition. Then type **2** to create the second primary partition, `/dev/hda2` in our case. When prompted for the first cylinder, hit enter. When prompted for the last cylinder, type **+512M** to create a partition 512MB in size. After you've done this, type **t** to set the partition type, **2** to select the partition you just created and then type in **82** to set the partition type to "Linux Swap". After completing these steps, typing **p** should display a partition table that looks similar to this:

Code listing: Our swap partition has been created

```
Command (m for help): p

Disk /dev/hda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot    Start       End    Blocks   Id  System
/dev/hda1          1        14    105808+  83  Linux
/dev/hda2         15        81    506520   82  Linux swap
```

Finally, let's create the root partition. To do this, type **n** to create a new partition, then **p** to tell fdisk that you want a primary partition. Then type **3** to create the third primary partition, `/dev/hda3` in our case. When prompted for the first cylinder, hit enter. When prompted for the last cylinder, hit enter to create a partition that takes up the rest of the remaining space on your disk. After completing these steps, typing **p** should display a partition table that looks similar to this:

Code listing: Our root partition has been created

```
Command (m for help): p

Disk /dev/hda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot    Start       End    Blocks   Id  System
/dev/hda1          1        14    105808+  83  Linux
/dev/hda2         15        81    506520   82  Linux swap
/dev/hda3         82      3876  28690200   83  Linux
```

Finally, we need to set the "bootable" flag on our boot partition and then write our changes to disk. To tag `/dev/hda1` as a "bootable" partition, type **a** at the menu and then type in **1** for the partition number. If you type **p** now, you'll now see that `/dev/hda1` has a ***** in the "Boot" column. Now, let's write our changes to disk. To do this, type **w** and hit enter. Your disk partitions are now properly configured for a Gentoo Linux install.

Note: If **`fdisk`** or **`cfdisk`** instruct you to do so, please reboot to allow your system to detect the new partition configuration.

Creating filesystems

Now that the partitions have been created, it's time to set up filesystems on the boot and root partitions so that they can be mounted and used to store data. We will also configure the swap partition to serve as swap storage.

Gentoo Linux supports a variety of different types of filesystems; each type has its strengths and weaknesses and its own set of performance characteristics. Currently, we support the creation of ext2, ext3, XFS, JFS and ReiserFS filesystems.

ext2 is the tried and true Linux filesystem but doesn't have metadata journaling, which means that routine ext2 filesystem checks at startup time can be quite time-consuming. There is now quite a selection of newer-generation **journaled** filesystems that can be checked for consistency very quickly and are thus generally preferred over their non-journaled counterparts. Journaled filesystems prevent long delays when you boot your system and your filesystem happens to be in an **inconsistent** state.

ext3 is the journaled version of the ext2 filesystem, providing metadata journaling for fast recovery in addition to other enhanced journaling modes like full data and ordered data journaling. ext3 is a very good and reliable filesystem. It offers generally decent performance under most conditions. Because it does not extensively employ the use of "trees" in its internal design, it doesn't scale very well, meaning that it is not an ideal choice for very large filesystems, or situations where you will be handling very large files or large quantities of files in a single directory. But when used within its design parameters, ext3 is an excellent filesystem.

ReiserFS is a B*-tree based filesystem that has very good overall performance and greatly outperforms both ext2 and ext3 when dealing with small files (files less than 4k), often by a factor of 10x-15x. ReiserFS also scales extremely well and has metadata journaling. As of kernel 2.4.18+, ReiserFS is now rock-solid and highly recommended for use both as a general-purpose filesystem and for extreme cases such as the creation of large filesystems, the use of many small files, very large files and directories containing tens of thousands of files. ReiserFS is the filesystem we recommend by default for all non-boot partitions.

XFS is a filesystem with metadata journaling that is fully supported under Gentoo Linux's **xfs-sources** kernel. It comes with a robust feature-set and is optimized for scalability. We only recommend using this filesystem on Linux systems with high-end SCSI and/or fibre channel storage and a uninterruptible power supply. Because XFS aggressively caches in-transit data in RAM, improperly designed programs (those that don't take proper precautions when writing files to disk and there are quite a few of them) can lose a good deal of data if the system goes down unexpectedly.

JFS is IBM's high-performance journaling filesystem. It has recently become production-ready and there hasn't been a sufficient track record to comment positively nor negatively on its general stability at this point.

If you're looking for the most rugged journaling filesystem, use ext3. If you're looking for a good general-purpose high-performance filesystem with journaling support, use ReiserFS; both ext3 and ReiserFS are mature, refined and recommended for general use.

Based on our example above, we will use the following commands to initialize all our partitions for use:

Code listing: Initializing our partitions (example)

```
# mke2fs -j /dev/hda1
# mkswap /dev/hda2
# mkreiserfs /dev/hda3
```

We choose ext3 for our `/dev/hda1` boot partition because it is a robust journaling filesystem supported by all major boot loaders. We used **mkswap** for our `/dev/hda2` swap partition -- the choice is obvious here. And for our main root filesystem on `/dev/hda3` we choose ReiserFS, since it is a solid journaling filesystem offering excellent performance. Now, go ahead and initialize your partitions.

For your reference, here are the various **mkfs**-like commands available during the installation process:

mkswap is the command that is used to initialize swap partitions:

Code listing: Initializing Swap

```
# mkswap /dev/hda2
```

You can use the **mke2fs** command to create ext2 filesystems:

Code listing: Creating an ext2 Filesystem

```
# mke2fs /dev/hda1
```

If you would like to use ext3, you can create ext3 filesystems using **mke2fs -j**:

Code listing: Creating an ext3 Filesystem

```
# mke2fs -j /dev/hda3
```

Note: You can find out more about using ext3 under Linux 2.4 at http://www.zip.com.au/~akpm/linux/ext3/ext3-usage.html.

To create ReiserFS filesystems, use the **mkreiserfs** command:

Code listing: Creating a ReiserFS Filesystem

```
# mkreiserfs /dev/hda3
```

To create an XFS filesystem, use the **mkfs.xfs** command:

Code listing: Creating a XFS Filesystem

```
# mkfs.xfs /dev/hda3
```

Note: You may want to add a couple of additional flags to the **mkfs.xfs** command: **-d agcount=3 -l size=32m**. The **-d agcount=3** command will

lower the number of allocation groups. XFS will insist on using at least 1 allocation group per 4 GB of your partition, so, for example, if you have a 20 GB partition you will need a minimum agcount of 5. The **-l size=32m** command increases the journal size to 32 Mb, increasing performance.

To create JFS filesystems, use the **mkfs.jfs** command:

Code listing: Creating a JFS Filesystem

```
# mkfs.jfs /dev/hda3
```

7. Mount Partitions

Now, we will activate our newly-initialized swap volume, since we may need the additional virtual memory that it provides later:

Code listing: Activating Swap

```
# swapon /dev/hda2
```

Next, we will create the `/mnt/gentoo/boot` mount point, and we will mount our filesystems to the mount points. Once our boot and root filesystems are mounted, any files we copy or create inside `/mnt/gentoo` will be placed on our new filesystems. Note that if you are setting up Gentoo Linux with separate `/usr` or `/var` filesystems, these would get mounted to `/mnt/gentoo/usr` and `/mnt/gentoo/var` respectively.

Note: If your `/boot` partition (the one holding the kernel) is ReiserFS, be sure to mount it with the **-o notail** option so GRUB gets properly installed. Make sure that **notail** ends up in your new `/etc/fstab` boot partition entry, too. We will get to that in a bit. If you are going to use LILO with ReiserFS, then the **-o notail** is not needed. It's always safe to specify the **-o notail** option with ReiserFS if you're not sure what to do.

Code listing: Creating Mount Points

```
# mount /dev/hda3 /mnt/gentoo
# mkdir /mnt/gentoo/boot
# mount /dev/hda1 /mnt/gentoo/boot
```

Note: If you are having problems mounting your boot partition with ext2, try using **mount /dev/hXX /mnt/gentoo/boot -t ext2**

8. Stage tarballs and chroot

Selecting the desired stage tarball

Now, you need to decide which one you would like to use as a basis for the install if you haven't already. The stages on the Live CD are in `/mnt/cdrom/stages/` and you can type **ls /mnt/cdrom/stages/** to see what's available on your CD.

GRP users should use the `stage3-xx-yy.tar.bz2` tarball.

If you would like to perform an install using a stage tarball that is **not** on your CD (which will likely be the case if you're using our "basic" Live CD), this is still possible, but you'll need to download the stage you want using the following instructions. If you already have the stage tarball you want to use (which most users will have), then proceed to the "Extracting the stage tarball" section.

Note: On the basic LiveCD, **lynx** is substituted with **links2**. To use **links2** with proxy, press Escape once inside **links2** and go to Setup, Network Options. Don't forget to save everything in Setup, Save Options.

Code listing: Downloading Required Stages

```
# cd /mnt/gentoo
Use lynx to get the URL for your tarball:
# lynx http://gentoo.oregonstate.edu/releases/x86/1.4/
Use Up and Down arrows keys (or the TAB key) to go to the right directory
Highlight the appropriate stage you want to download
Press d which will initiate the download
Save the file and quit the browser

OR use wget from the command line:
# wget (insert URL to the required stage tarball here)
```

Extracting the stage tarball

Now it is time to extract the compressed stage tarball of your choice to `/mnt/gentoo/`. Remember, you only need to unpack one stage tarball, either a stage1, stage2 or stage3. So, if you wanted to perform a stage3 install of Gentoo, then you would just unpack the stage3 tarball. Unpack the stage tarball as follows:

Note: Be sure to use the **p** option with **tar**. Forgetting to do this will cause certain files to have incorrect permissions.

Code listing: Unpacking the Stages

```
# cd /mnt/gentoo
Change "stage3" to "stage2" or "stage1" if you want to start from these stages instead.
If you downloaded your stage tarball, change the path below to begin with "/mnt/gentoo/"
instead of "/mnt/cdrom/stages/".
# tar -xvjpf /mnt/cdrom/stages/stage3-*.tar.bz2
```

If you downloaded your stage tarball to `/mnt/gentoo`, you can now delete it by typing **rm /mnt/gentoo/stage*.tar.bz2**.

GRP package/snapshot steps

Note: The following instructions are for GRP users only.

GRP Users: There is a Portage snapshot on the Live CD. You will need to use this snapshot so that you can skip the **emerge sync** step later in this document, since **emerge sync** requires a network connection. Untar this snapshot as follows:

Code listing: Using Portage snapshot

```
Replace yyyymmdd with the datestamp in the filename.
# tar -xvjf /mnt/cdrom/snapshots/portage-yyyymmdd.tar.bz2 -C /mnt/gentoo/usr
```

This will extract a snapshot of the Portage tree to your fresh Gentoo install. Now you won't need to connect to the Internet and use **emerge sync** to download a Portage tree. Now, copy distfiles and packages from the Live CD into place:

Code listing: Copying GRP files

```
# cp -R /mnt/cdrom/distfiles /mnt/gentoo/usr/portage/distfiles
# cp -a /mnt/cdrom/packages /mnt/gentoo/usr/portage/packages
```

All relevant files are now in place for using GRP. You should now have everything copied over and unpacked that you'll need to install Gentoo Linux -- even without a network connection.

Selecting Mirrors (Optional)

mirrorselect is a tool designed to automatically pick the fastest mirrors based on your location, or manually pick a mirror from a list. Unfortunately, **mirrorselect** does not work well behind all routers.

Code listing: Using mirrorselect

```
To select a mirror automatically:
# mirrorselect -a -s4 -o >> /mnt/gentoo/etc/make.conf
To select a mirror interactively:
# mirrorselect -i -o >> /mnt/gentoo/etc/make.conf
```

If for some reason **mirrorselect** fails you should be able to continue with this guide since no changes are made.

Entering the chroot

Next, we will **chroot** over to the new Gentoo Linux build installation to "enter" the new Gentoo Linux system:

Note: You may receive a notice during **env-update** telling you that `/etc/make.profile/make.defaults` isn't available: ignore it. We are going to issue **emerge sync** later on in this document, which will resolve the problem.

Code listing: Prepping and entering the chroot environment

```
# mount -t proc proc /mnt/gentoo/proc
# cp /etc/resolv.conf /mnt/gentoo/etc/resolv.conf
# chroot /mnt/gentoo /bin/bash
# env-update
Regenerating /etc/ld.so.cache...
# source /etc/profile
(The above points your shell to the new paths and updated binaries)
```

After you execute these commands, you will be "inside" your new Gentoo Linux environment in `/mnt/gentoo`. We can perform the rest of the installation process inside the chroot.

9. Getting the Current Portage Tree using sync

Note: If you are doing a GRP install then you can ignore the following section on **emerge sync**.

Now, you will need to run **emerge sync**. This command tells Portage to download the most recent copy of the Gentoo Linux Portage tree from the Internet. If you extracted a Portage tree snapshot from **CD 1** earlier, you can safely skip this step. The Portage tree contains all the scripts (called ebuilds) used to build every package under Gentoo Linux. Currently, we have ebuild scripts for close to 4000 packages. Once **emerge sync** completes, you will have a complete Portage tree in `/usr/portage`:

Code listing: Updating Using sync

```
# emerge sync
```

10. Setting Gentoo optimizations (make.conf)

Now that you have a working copy of the Portage tree, it is time to customize the optimization and optional build-time settings to use on your Gentoo Linux system. Portage will use these settings when compiling any programs for you. To do this, edit the file `/etc/make.conf`. In this file, you should set your USE flags, which specify optional functionality that you would like to be built into packages if available; generally, the defaults (an **empty** or unset USE variable) are fine. More information on USE flags can be found here. A complete list of current USE flags can be found here.

If you are starting from a stage1 tarball, You also should set appropriate CHOST, CFLAGS and CXXFLAGS settings for the kind of system that you are creating (commented examples can be found further down in the file). If you are using a stage2 or stage3 tarball, these settings will already be configured

optimally and should not require any modification.

Note: Advanced users: If you are planning on installing an ACCEPT_KEYWORDS="~x86" Gentoo system, do not set ACCEPT_KEYWORDS until the bootstrap phase (stage1) is done.

Note: Advanced users: The CFLAGS and CXXFLAGS settings are used to tell the C and C++ compiler how to optimize the code that is generated on your system. It is common for users with Athlon XP processors to specify a "-march=athlon-xp" setting in their CFLAGS and CXXFLAGS settings so that all packages built will be optimized for the instruction set and performance characteristics of their CPU, for example. The `/etc/make.conf` file contains a general guide for the proper settings of CFLAGS and CXXFLAGS.

If necessary, you can also set proxy information here if you are behind a firewall. Use the following command to edit `/etc/make.conf` using **nano**, a simple visual editor:

Code listing: Setting make.conf Options

```
# nano -w /etc/make.conf
```

Note: Advanced users: People who need to substantially customize the build process should take a look at the `/etc/make.globals` file. This file comprises gentoo defaults and should never be touched. If the defaults do not suffice, then new values should be put in `/etc/make.conf`, as entries in `make.conf` **override** the entries in `make.globals`. If you're interested in customizing USE settings, look in `/etc/make.profile/make.defaults`. If you want to turn off any USE settings found here, add an appropriate **USE="-foo"** in `/etc/make.conf` to turn off any **foo** USE setting enabled by default in `/etc/make.globals` or `/etc/make.profile/make.defaults`.

Note: Make sure not to add '**static**' to your USE variables until after stage1.

11. Starting from Stage1

Note: If you are not starting from a stage1 tarball, skip this section.

The stage1 tarball is for complete customization and optimization. If you have picked this tarball, you are most likely looking to have an uber-optimized and up-to-date system. Have fun! Installing from a stage1 takes a lot of time, but the result is a system that has been optimized from the ground up for your specific machine and needs.

Now, it is time to start the "bootstrap" process. This process takes about two hours on a 1200MHz AMD Athlon system. During this time, the GNU C library, compiler suite and other key system programs will be built. Start the bootstrap as follows:

Code listing: Bootstrapping

```
# cd /usr/portage
# scripts/bootstrap.sh
```

The "bootstrap" process will now begin.

Note: `bootstrap.sh` now supports the **`--fetchonly`** option. Dial-up users will find this especially handy. It will download all bootstrap related files in one go for later compilation. See **`bootstrap.sh -h`** for more information.

Note: Portage by default uses `/var/tmp` during package building, often using several hundred megabytes of temporary storage. If you would like to change where Portage stores these temporary files, set a new PORTAGE_TMPDIR **before** starting the bootstrap process, as follows:

Code listing: Changing Portage's Storage Path

```
# export PORTAGE_TMPDIR="/otherdir/tmp"
```

`bootstrap.sh` will build **`binutils`**, **`gcc`**, **`gettext`**, and **`glibc`**, rebuilding **`gettext`** after **`glibc`**. Needless to say, this process takes a while. Once this process completes, your system will be equivalent to a "stage2" system, which means you can now move on to the stage2 instructions.

12. Starting from Stage2 and continuing Stage1

Note: This section is for those continuing a stage1 install or starting at stage2. If this is not you (ie. you're using a stage3), then skip this section.

Note: If you start from stage2, don't change the CHOST variable in `/etc/make.conf`. Doing so results in strange and broad compilation failures.

The stage2 tarball already has the bootstrapping done for you. All that you have to do is install the rest of the system:

Note: If you are starting from a pre-built stage2 and want to ensure that your compiler toolchain is fully up-to-date, add the **`-u`** option to the commands below. If you don't know what this means, it's safe to skip this suggestion.

Code listing: Installing the rest of the system

```
# emerge -p system
(lists the packages to be installed)
# emerge system
```

It is going to take a while to finish building the entire base system. Your reward is that it will be thoroughly optimized for your system. The drawback is that you have to find a way to keep yourself occupied for some time to come. The author suggests "Star Wars - Super Bombad Racing" for the PS2.

Building is now complete. Go ahead and skip down to the "Setting your time zone" section.

13. Starting from Stage3

Note: This section is for those starting with stage3 and not for those who have started with stage1 or stage2 who should skip this section. GRP users should skip ahead to the next section.

Note: Remember, if you start from stage3, don't change the CHOST variable in `/etc/make.conf`. Doing so can result in compilation failures.

The stage3 tarball provides a fully-functional basic Gentoo system, so no building is required.

Note: Advanced users: However, since the stage3 tarball is pre-built, it may be slightly out-of-date. If this is a concern for you, you can automatically update your existing stage3 to contain the most up-to-date versions of all system packages by making a backup of `/etc/make.conf`, then typing **`export CONFIG_PROTECT="-*" emerge -u system`** (this requires a network connection) and replacing the backup afterwards. Note that this could take a long time if your stage3 is very old; otherwise, this process will generally be quick and will allow you to benefit from the very latest Gentoo updates and fixes. In any case, feel free to skip these steps and proceed to the next section if you like.

14. Setting your time zone

Now you need to set your time zone.

Look for your time zone (or GMT if you are using Greenwich Mean Time) in `/usr/share/zoneinfo`. Then, make a symbolic link to `/etc/localtime` by typing:

Code listing: Creating a symbolic link for time zone

```
# ln -sf /usr/share/zoneinfo/path/to/timezonefile /etc/localtime
```

15. Modifying /etc/fstab for your machine

Note: To edit files, remember to use **`nano -w "filename"`**.

Your Gentoo Linux system is almost ready for use. All we need to do now is configure a few important system files and install the boot loader. The first file we need to configure is `/etc/fstab`. Remember that you should use the **`notail`** option for your boot partition if you chose to create a ReiserFS filesystem on it. Remember to specify **`ext2`**, **`ext3`** or **`reiserfs`** filesystem types as appropriate.

Note: Use something like the `/etc/fstab` listed below, but of course be sure to replace "BOOT", "ROOT" and "SWAP" with the actual block devices (such as **`hda1`**, etc.) and "ext2" and "ext3" with the actual filesystems you are using:

Code listing: Editing fstab

```
# /etc/fstab: static file system information.
#
# noatime turns off atimes for increased performance (atimes normally aren't
# needed; notail increases performance of ReiserFS (at the expense of storage
# efficiency).  It is safe to drop the noatime options if you want and to
# switch between notail and tail freely.

# <fs>           <mount point>   <type>   <opts>          <dump/pass>

# NOTE: If your BOOT partition is ReiserFS, add the notail option to opts.

/dev/BOOT           /boot       ext2  noauto,noatime  1 2
/dev/ROOT           /           ext3  noatime         0 1
/dev/SWAP           none        swap  sw              0 0
/dev/cdroms/cdrom0  /mnt/cdrom  iso9660  noauto,ro       0 0
proc                /proc       proc  defaults        0 0
```

Note: Please notice that `/boot` is **not** mounted at boot time. This is to protect the data in `/boot` from corruption. If you need to access `/boot`, please mount it!

16. Installing the kernel and system logger

Kernel selections

There are two options for installing a kernel. You can either configure your own kernel or use the **`genkernel`** utility to configure and compile your kernel automatically.

Whether configuring a kernel by hand or using **`genkernel`**, you'll need to merge the Linux kernel sources you'd like to use. Gentoo provides several kernel ebuilds; a list can be found here. If you are uncertain which kernel sources to choose, we advise using **`gentoo-sources`**. If you want XFS support, you should choose **`xfs-sources`** or **`gs-sources`**. Gentoo's LiveCD uses **`gs-sources`** and **`xfs-sources`**. There is also a **`gaming-sources`** kernel optimized for game-playing responsiveness that works wonderfully for this purpose when the "Preemptible kernel" option is enabled.

Choose a kernel and then merge as follows:

Code listing: Emerging Kernel Sources

```
# emerge -k sys-kernel/gentoo-sources
```

The `/usr/src/linux` symbolic link will point to your newly-installed kernel source tree. Portage uses the `/usr/src/linux` symbolic link for a special purpose. Any ebuilds you install that contain kernel modules will be configured to work with the kernel source tree pointed to by `/usr/src/linux`. `/usr/src/linux` is created when you emerge your first kernel source package, but after it exists, Portage does not modify this symbolic link.

Using genkernel to compile your kernel

Now that your kernel source tree is installed, it's now time to compile your kernel. There are two ways to do this. The first way is to use our new **genkernel** script to automatically build a kernel for you. **genkernel** works by configuring a kernel nearly identically to the way our LiveCD kernel is configured. This means that when you use **genkernel** to build your kernel, your system will generally detect all your hardware at boot-time, just like our Live CD does. Because genkernel doesn't require any manual kernel configuration, it is an ideal solution for those users who may not be comfortable compiling their own kernels.

Now, let's see how to use genkernel. First, emerge the genkernel ebuild:

Code listing: Emerging genkernel

```
# emerge -k genkernel
```

Now, compile your kernel sources by running **genkernel**:

Note: Advanced users: you can type **genkernel --config** instead, which will cause genkernel to allow you to tweak the default kernel configuration before building begins.

Code listing: Running genkernel

```
If you're using genkernel 1.2 (included in the 1.4-20030803 x86/i686 GRP set), use the following:
# genkernel gentoo-sources
If you're using genkernel 1.4 or newer, there's no need to specify a kernel:
# genkernel
Gentoo Linux genkernel, version 1.4
Copyright 2003 Gentoo Technologies, Inc., Bob Johnson, Daniel Robbins
Distributed under the GNU General Public License version 2

Settings:
compile optimization: 1 processor(s)
source tree: /usr/src/linux-2.4.20-gaming-r3
config: gentoo (customized)
config loc: /etc/kernels/config-2.4.20-gaming-r3
initrd config: (default) /etc/kernels/settings

* Running "make oldconfig"...                                                    [ ok ]
* Logging to /var/log/genkernel.log...                                          [ ok ]
* Starting 2.4.20-gaming-r3 build...                                            [ ok ]
* Running "make dep"...                                                         [ ok ]
* Running "make bzImage"...                                                     [ ok ]
* Running "make modules"...                                                     [ ok ]
* Running "make modules_install"...                                             [ ok ]
* Moving bzImage to /boot/kernel-2.4.20-gaming-r3...                            [ ok ]
* Building busybox...                                                           [ ok ]
* Creating initrd...                                                            [ ok ]

* Build completed successfully!

* Please specify /boot/kernel-2.4.20-gaming-r3 and /boot/initrd-2.4.20-gaming-r3
* when customizing your boot loader configuration files.
```

Once **genkernel** completes, a kernel, full set of modules and **initial root disk** (initrd) will be created. We will use the kernel and initrd when configuring a boot loader later in this document. Write down the names of the kernel and initrd as you will need it when writing the bootloader configuration file. The initrd will be started immediately after booting to perform hardware autodetection (just like on the Live CD) before your "real" system starts up.

Now, let's perform one more step to get our system to be more like the Live CD -- let's emerge **hotplug**. While the initrd autodetects hardware that is needed to boot your system, **hotplug** autodetects everything else. To emerge and enable **hotplug**, type the following:

Code listing: Emerging and enabling hotplug

```
# emerge -k hotplug
# rc-update add hotplug default
```

Now that you've run and configured your system to use **genkernel**, you can skip the "manual kernel configuration" section below.

Manual kernel configuration

If you opted not to use genkernel to compile your kernel, this section will guide you through the process of configuring and compiling a kernel by hand. Please note that `/usr/src/linux` is a symlink to your current emerged kernel source package and is set automatically by Portage at emerge time. If you have multiple kernel source packages, it is necessary to set the `/usr/src/linux` symlink to the correct one before proceeding.

Note: If you are configuring your own kernel, be careful with the *grsecurity* option. Being too aggressive with your security settings can cause certain programs (such as X) to not run properly. If in doubt, leave it out.

Note: If you want to use the same configuration as the LiveCD kernel or base your configuration on it, you should execute **cd /usr/src/linux && cat /proc/config > .config && make oldconfig**. If you aren't using **xfs-sources**, this will ask some questions about differences between your kernelchoice and **xfs-sources**.

Code listing: Configuring the Linux Kernel

```
# cd /usr/src/linux
```

```
# make menuconfig
```

Note: For your kernel to function properly, there are several options that you will need to ensure are in the kernel proper -- that is, they should **be enabled and not compiled as modules**. Be sure to enable "ReiserFS" if you have any ReiserFS partitions; the same goes for "Ext3". If you're using XFS, enable the "SGI XFS filesystem support" option. It's always a good idea to leave ext2 enabled whether you are using it or not.

Below are some common options that you will need:

Code listing: make menuconfig options

```
Code maturity level options --->
[*] Prompt for development and/or incomplete code/drivers"
(You need this to enable some of the options below)
...

File systems --->
<*> Reiserfs support
(Only needed if you are using reiserfs)
...
<*> Ext3 journalling file system support
(Only needed if you are using ext3)
...
[*] Virtual memory file system support (former shm fs)
(Required for Gentoo Linux)
...
<*> JFS filesystem support
(Only needed if you are using JFS)
...
[*] /proc file system support
(Required for Gentoo Linux)
[*] /dev file system support (EXPERIMENTAL)
[*]   Automatically mount at boot
(Required for Gentoo Linux)
[ ] /dev/pts file system for Unix98 PTYs
(Uncheck this, it is not needed unless you use a 2.6 kernel)
...
<*> Second extended fs support
(Only needed if you are using ext2)
...
<*> XFS filesystem support
(Only needed if you are using XFS)
```

If you use PPPoE to connect to Internet, you will need the following options in the kernel (built-in or as preferably as modules) : "PPP (point-to-point protocol) support", "PPP support for async serial ports", "PPP support for sync tty ports". The two compression options won't harm but are not definitely needed, neither does the "PPP over Ethernet" option, that might only be used by **`rp-pppoe`** when configured to do kernel mode PPPoE.

If you have an IDE cd burner, then you need to enable SCSI emulation in the kernel. Turn on "ATA/IDE/MFM/RLL support" ---> "IDE, ATA and ATAPI Block devices" ---> "SCSI emulation support" (I usually make it a module), then under "SCSI support" enable "SCSI support", "SCSI CD-ROM support" and "SCSI generic support" (again, I usually compile them as modules). If you also choose to use modules, then **`echo -e "ide-scsi\nsg\nsr_mod" >> /etc/modules.autoload.d/kernel-2.4`** to have them automatically added at boot time.

If you require it, don't forget to include support in the kernel for your ethernet card.

Note: For those who prefer it, it is possible to install Gentoo Linux with a 2.2 kernel. However, doing this comes at a price: you will lose many of the nifty features that are new to the 2.4 series kernels (such as XFS and tmpfs filesystems, iptables and more), although the 2.2 kernel sources can be patched with ReiserFS and devfs support. Gentoo linux boot scripts require either tmpfs or ramdisk support in the kernel, so 2.2 kernel users need to make sure that ramdisk support is compiled in (ie, not a module). It is vital that a **gentoo=notmpfs** flag be added to the kernel line in `/boot/grub/grub.conf` or to the append line in `/etc/lilo.conf` for the 2.2 kernel so that a ramdisk is mounted for the boot scripts instead of tmpfs. If you choose not to use devfs, then **gentoo=notmpfs,nodevfs** should be used instead.

Code listing: Compiling and Installing the kernel

```
# make dep && make clean bzImage modules modules_install
# cp /usr/src/linux/arch/i386/boot/bzImage /boot
```

Installing additional hardware-specific ebuilds

Finally, you should emerge ebuilds for any additional hardware that is on your system. Here is a list of kernel-related ebuilds that you could emerge:

Ebuild	Purpose	Command
nvidia-kernel	Accelerated NVIDIA graphics for XFree86	**`emerge -k nvidia-kernel`**
nforce-net	On-board ethernet controller on NVIDIA NForce(2) motherboards	**`emerge nforce-net`**
nforce-audio	On-board audio on NVIDIA NForce(2) motherboards	**`emerge nforce-audio`**
e100	Intel e100 Fast Ethernet Adapters	**`emerge e100`**
e1000	Intel e1000 Gigabit Ethernet Adapters	**`emerge e1000`**
emu10k1	Creative Sound Blaster Live!/Audigy support	**`emerge emu10k1`**
ati-drivers	Accelerated ATI Radeon 8500+/FireGL graphics for XFree86	**`emerge ati-drivers`**
xfree-drm	Accelerated graphics for ATI Radeon up to 9200, Rage128, Matrox, Voodoo and other cards for XFree86	**`VIDEO_CARDS="yourcard" emerge xfree-drm`**

The **`nvidia-kernel`**, **`ati-drivers`** and **`xfree-drm`** packages will require additional configuration to be enabled. All other ebuilds listed above should be

auto-detected at boot-time by the **hotplug** package. If you are not using hotplug, be sure to add the appropriate modules to /etc/modules.autoload.d/kernel-2.4.

More information on **xfree-drm** can be found in our Direct Rendering Guide.

Installing a system logger

Your new custom kernel (and modules) are now installed. Now you need to choose a system logger that you would like to install. We offer sysklogd, which is the traditional set of system logging daemons. We also have msyslog and syslog-ng as well as metalog. Power users seem to gravitate away from sysklogd (not very good performance) and towards the newer alternatives. If in doubt, you may want to try metalog, since it seems to be quite popular. To merge your logger of choice, type **one** of the next four command sets:

Code listing: Emerging System Logger of Choice

```
# emerge -k app-admin/sysklogd
# rc-update add sysklogd default
or
# emerge -k app-admin/syslog-ng
# rc-update add syslog-ng default
or
# emerge -k app-admin/metalog
# rc-update add metalog default
or
# emerge -k app-admin/msyslog
# rc-update add msyslog default
```

Note: Metalog flushes output to the disk in blocks, so messages aren't immediately recorded into the system logs. If you are trying to debug a daemon, this performance-enhancing behavior is less than helpful. When your Gentoo Linux system is up and running, you can send metalog a USR1 signal to temporarily turn off this message buffering (meaning that **tail -f /var/log/everything/current** will now work in real time, as expected) and a USR2 signal to turn buffering back on again. If you want to disable buffering permanently, you can change METALOG_OPTS="-B" to METALOG_OPTS="-B -s" in /etc/conf.d/metalog.

Code listing: Turning metalog buffering on/off

```
To turn the buffering off:
# killall -USR1 metalog
To turn the buffering back on:
# killall -USR2 metalog
```

Now, you may optionally choose a cron package that you would like to use. Right now, we offer dcron, fcron and vcron. If you do not know which one to choose, you might as well grab vcron.

Code listing: Choosing a CRON Daemon

```
# emerge -k sys-apps/dcron
# rc-update add dcron default
# crontab /etc/crontab
or
# emerge -k sys-apps/fcron
# rc-update add fcron default
# crontab /etc/crontab
or
# emerge -k sys-apps/vcron
# rc-update add vcron default
You do not need to run crontab /etc/crontab if using vcron.
```

For more information on starting programs and daemons at startup, see the rc-script guide.

17. Installing miscellaneous necessary packages

If you need rp-pppoe to connect to the net, be aware that at this point it has not been installed. It would be the good time to do it:

Code listing: Installing rp-pppoe

```
# USE="-X" emerge rp-pppoe
GRP users should type the following:
# emerge -K rp-pppoe
```

Note: The *USE="-X"* prevents pppoe from installing its optional X interface, which is a good thing, because X and its dependencies would also be emerged. You can always recompile *rp-pppoe* with X support later. The GRP version of rp-pppoe has the optional X interface enabled. If you're not using GRP, compile from source as in the first example.

Note: Please note that the rp-pppoe is built but not configured. You will have to do it again using **adsl-setup** when you boot into your Gentoo system for the first time.

You may need to install some additional packages in the Portage tree if you are using any optional features like XFS, ReiserFS or LVM. If you're using XFS, you should emerge the **xfsprogs** package:

Code listing: Emerging Filesystem Tools

```
# emerge -k sys-apps/xfsprogs
If you would like to use ReiserFS, you should emerge the ReiserFS tools:
# emerge -k sys-apps/reiserfsprogs
If you would like to use JFS, you should emerge the JFS tools:
# emerge -k jfsutils
If you're using LVM, you should emerge the lvm-user package:
# emerge -k sys-apps/lvm-user
```

If you're a laptop user and wish to use your PCMCIA slots on your first real reboot, you will want to make sure you install the *pcmcia-cs* package.

Code listing: Emerging PCMCIA-cs

```
# emerge -k sys-apps/pcmcia-cs
```

18. User Management

Setting a root password

Before you forget, set the root password by typing:

Code listing: Setting the root Password

```
# passwd
```

Adding a user for day-to-day use

Working as root on a Unix/Linux system is **dangerous** and should be avoided as much as possible. Therefor it is **strongly** recommended to add a user for day-to-day use:

Code listing: Adding a user

```
# useradd your_user -m -G users,wheel,audio -s /bin/bash
# passwd your_user
```

Substitute **your_user** with your username.

Whenever you need to perform some task that only root can handle, use **su -** to change your privileges to root-privileges, or take a look at the **sudo** package.

19. Setting your Hostname

Edit `/etc/hostname` so that it contains your hostname on a single line, i.e. **mymachine**.

Code listing: Configuring Hostname

```
# echo mymachine > /etc/hostname
```

Then edit `/etc/dnsdomainname` so that it contains your DNS domainname, i.e. **mydomain.com**.

Code listing: Configuring Domainname

```
# echo mydomain.com > /etc/dnsdomainname
```

If you have a NIS domain, you should set it in `/etc/nisdomainname`.

Code listing: Configuring NIS Domainname

```
# echo nis.mydomain.com > /etc/nisdomainname
```

20. Modifying /etc/hosts

This file contains a list of IP addresses and their associated hostnames. It is used by the system to resolve the IP addresses of any hostnames that may not be in your nameservers. Here is a template for this file:

Code listing: Hosts Template

```
127.0.0.1      localhost
# the next line contains your IP for your local LAN and your associated machine name
192.168.1.1    mymachine.mydomain.com mymachine
```

Note: If you are on a DHCP network, it might be helpful to add your machine's actual hostname after *localhost*. This will help GNOME and many other programs in name resolution.

21. Final Network Configuration

Add the names of any modules that are necessary for the proper functioning of your system to `/etc/modules.autoload.d/kernel-2.4` file (you can also add any options you need to the same line). When Gentoo Linux boots, these modules will be automatically loaded. Of particular importance is your

ethernet card module, if you happened to compile it as a module:

Code listing: /etc/modules.autoload.d/kernel-2.4

```
This is assuming that you are using a 3com card.
Check /lib/modules/`uname -r`/kernel/drivers/net for your card.
3c59x
```

Edit the `/etc/conf.d/net` script to get your network configured for your first boot:

Code listing: Boot time Network Configuration

```
# nano -w /etc/conf.d/net
Only for non-PCMCIA network cards:
# rc-update add net.eth0 default
```

If you have multiple network cards or tokenring interfaces, you need to create additional `net.ethx` or `net.trx` scripts respectively for each one (x = 1, 2, ...):

Code listing: Multiple Network Interfaces

```
# cd /etc/init.d
# cp net.eth0 net.ethx
Only for non-PCMCIA network cards:
# rc-update add net.ethx default
```

If you have a PCMCIA card installed, have a quick look into `/etc/init.d/pcmcia` to verify that things seem all right for your setup, then run the following command:

Code listing: Have PCMCIA services start automatically

```
# rc-update add pcmcia boot
```

This makes sure that the PCMCIA drivers are autoloaded whenever your network is loaded. The appropriate `/etc/init.d/net.eth*` services will be started by the pcmcia service automatically.

22. Final steps: Configure Basic Settings (including the international keymap setting)

Code listing: Basic Configuration

```
# nano -w /etc/rc.conf
```

Follow the directions in the file to configure the basic settings. All users will want to make sure that CLOCK is set to his/her liking. International keyboard users will want to set the KEYMAP variable (browse `/usr/share/keymaps` to see the various possibilities).

23. Configure a Bootloader

Notes

In the spirit of Gentoo, users now have more than one bootloader to choose from. Using our virtual package system, users are now able to choose between both GRUB and LILO as their bootloaders.

Please keep in mind that having both bootloaders installed is not necessary. In fact, it can be a hindrance, so please only choose one.

In addition, you will need to configure our bootloader differently depending upon whether you are using **`genkernel`** (with kernel and initrd) or a kernel you compiled by hand. Be sure to take note of the important differences.

Note: If you are installing Gentoo Linux on a system with an NVIDIA nForce or nForce2 chipset with an integrated GeForce graphics card, you should use LILO and avoid GRUB. With on-board video enabled, the low memory area of your RAM may be used as video RAM. Since GRUB also uses low memory at boot time, it may experience an "out of memory" condition. So, if you have an nForce or potentially other board with on-board video, use LILO. Even if you're using off-board video right now, it would be nice to be able to remove the graphics card and use the on-board video in a pinch, wouldn't it? :)

Configuring GRUB

The most critical part of understanding GRUB is getting comfortable with how GRUB refers to hard drives and partitions. Your Linux partition `/dev/hda1` is called `(hd0,0)` under GRUB. Notice the parenthesis around the hd0,0 - they are required. Hard drives count from zero rather than "a" and partitions start at zero rather than one. Be aware too that with the hd devices, only hard drives are counted, not atapi-ide devices such as cdrom players, burners and that the same construct can be used with scsi drives. (Normally they get higher numbers than ide drives except when the bios is configured to boot from scsi devices.) Assuming you have a hard drive on `/dev/hda`, a cdrom player on `/dev/hdb`, a burner on `/dev/hdc`, a second hard drive on `/dev/hdd` and no SCSI hard drive, `/dev/hdd7` gets translated to `(hd1,6)`. It might sound tricky and tricky it is indeed, but as we will see, GRUB offers a tab completion mechanism that comes handy for those of you having a lot of hard drives and partitions and who are a little lost in the GRUB numbering scheme. Having gotten the feel for that, it is time to install GRUB.

The easiest way to install GRUB is to simply type **`grub`** at your chrooted shell prompt:

Code listing: Installing GRUB

```
# emerge -k grub
```

```
# grub
```

You will be presented with the **grub>** grub command-line prompt. Now, you need to type in the right commands to install the GRUB boot record onto your hard drive. In my example configuration, I want to install the GRUB boot record on my hard drive's MBR (master boot record), so that the first thing I see when I turn on the computer is the GRUB prompt. In my case, the commands I want to type are:

Code listing: GRUB on the MBR

```
grub> root (hd0,0) (Your boot partition)
grub> setup (hd0) (Where the boot record is installed; here, it is the MBR)
```

Code listing: GRUB not on the MBR

```
Alternatively, if you wanted to install the bootloader somewhere other than the MBR:
grub> root (hd0,0) (Your boot partition)
grub> setup (hd0,4) (Where the boot record is installed; here it is /dev/hda5)
grub> quit
```

Here is how the two commands work. The first **root ( )** command tells GRUB the location of your boot partition (in our example, `/dev/hda1` or `(hd0,0)` in GRUB terminology. Then, the second **setup ( )** command tells GRUB where to install the boot record - it will be configured to look for its special files at the **root ( )** location that you specified. In my case, I want the boot record on the MBR of the hard drive, so I simply specify `/dev/hda` (also known as `(hd0)`). If I were using another boot loader and wanted to set up GRUB as a secondary boot-loader, I could install GRUB to the boot record of a particular partition. In that case, I would specify a particular partition rather than the entire disk. Once the GRUB boot record has been successfully installed, you can type **quit** to quit GRUB.

Note: The tab completion mechanism of GRUB can be used from within GRUB, assuming you wrote **root (** and that you hit the TAB key, you would be prompted with a list of the available devices (not only hard drives), hitting the TAB key having written **root (hd**, GRUB would print the available hard drives and hitting the TAB key after writing **root (hd0,** would make GRUB print the list of partitions on the first hard drive. Checking the syntax of the GRUB location with completion should really help to make the right choice.

Gentoo Linux is now installed, but we need to create the `/boot/grub/grub.conf` file so that we get a nice GRUB boot menu when the system reboots. Here is how to do it.

Note: To ensure backwards compatibility with GRUB, make sure to make a link from `grub.conf` to `menu.lst`. You can do this by typing **ln -s /boot/grub/grub.conf /boot/grub/menu.lst**.

Now, create the `grub.conf` file (**nano -w /boot/grub/grub.conf**) and add the following to it:

Code listing: grub.conf for GRUB

```
default 0
timeout 30
splashimage=(hd0,0)/boot/grub/splash.xpm.gz

# If you compiled your own kernel, use something like this:
title=My example Gentoo Linux
root (hd0,0)
kernel (hd0,0)/boot/bzImage root=/dev/hda3

# If you're using genkernel, use something like this instead:
title=My example Gentoo Linux (genkernel)
root (hd0,0)
kernel (hd0,0)/boot/kernel-KV root=/dev/hda3
initrd (hd0,0)/boot/initrd-KV

# Below needed only for people who dual-boot
title=Windows XP
root (hd0,5)
chainloader (hd0,5)+1
```

Note: Substitute **KV** with the kernel version you have installed.

Note: (hd0,0) should be written without any spaces inside the parentheses.

Note: If you set up SCSI emulation for an IDE cd burner earlier, then to get it to actually work you need to add an **hdx=ide-scsi** fragment to the kernel line in `grub.conf` (where "hdx" should be the device for your cd burner).

After saving this file, Gentoo Linux installation is complete. Selecting the first option will tell GRUB to boot Gentoo Linux without a fuss. The second part of the `grub.conf` file is optional and shows you how to use GRUB to boot a bootable Windows partition.

Note: Above, `(hd0,0)` should point to your "boot" partition (`/dev/hda1` in our example config) and `/dev/hda3` should point to your root filesystem. `(hd0,5)` contains the NT boot loader.

Note: The path to the kernel image is relative to the boot partition. If for example you have separated boot partition `(hd0,0)` and root partition `(hd0,1)`, all paths in the `grub.conf` file above will become `/bzImage`.

If you need to pass any additional options to the kernel, simply add them to the end of the **kernel** command. We're already passing one option (**root=/dev/hda3**), but you can pass others as well. In particular, you can turn off devfs by default (not recommended unless you know what you're doing) by adding the **gentoo=nodevfs** option to the **kernel** command.

Note: Unlike in earlier versions of Gentoo Linux, you no longer have to add **devfs=mount** to the end of the **kernel** line to enable devfs. Now devfs is

enabled by default.

Configuring LILO

While GRUB may be the new alternative for most people, it is not always the best choice. LILO, the LInuxLOader, is the tried and true workhorse of Linux bootloaders. Here is how to install LILO if you would like to use it instead of GRUB.

The first step is to emerge LILO:

Code listing: Emerging LILO

```
# emerge -k lilo
```

Now it is time to configure LILO. Here is a sample configuration file `/etc/lilo.conf`:

Code listing: Example lilo.conf

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
lba32
default=linux

# Use something like the following 4 lines if you compiled your kernel yourself
image=/boot/bzImage
        label=linux
        read-only
        root=/dev/hda3

# If you used genkernel, use something like this:
image=/boot/kernel-KV
        label=gk_linux
        root=/dev/hda3
        initrd=/boot/initrd-KV
        append="root=/dev/hda3 init=/linuxrc"


# For dual booting windows/other OS
other=/dev/hda1
        label=dos
```

Note: Substitute **KV** with the kernel version you have installed, and make sure that **default=** points to your label (**gk_linux** if you used genkernel).

boot=/dev/hda tells LILO to install itself on the first hard disk on the first IDE controller. **map=/boot/map** states the map file. In normal use, this should not be modified. **install=/boot/boot.b** tells LILO to install the specified file as the new boot sector. In normal use, this should not be altered. If the install line is missing, LILO will assume a default of `/boot/boot.b` as the file to be used. The existence of **prompt** tells LILO to display the classic **lilo:** prompt at bootup. While it is not recommended that you remove the prompt line, if you do remove it, you can still get a prompt by holding down the [Shift] key while your machine starts to boot. **timeout=50** sets the amount of time that LILO will wait for user input before proceeding with booting the default line entry. This is measured in tenths of a second, with 50 as the default. **lba32** describes the hard disk geometry to LILO. Another common entry here is linear. You should not change this line unless you are very aware of what you are doing. Otherwise, you could put your system in an unbootable state. **default=linux** refers to the default operating system for LILO to boot from the options listed below this line. The name linux refers to the label line below in each of the boot options. **image=/boot/bzImage** specifies the linux kernel to boot with this particular boot option. **label=linux** names the operating system option in the LILO screen. In this case, it is also the name referred to by the default line. **read-only** specifies that the root partition (see the root line below) is read-only and cannot be altered during the boot process. **root=/dev/hda3** tells LILO what disk partition to use as the root partition.
After you have edited your `lilo.conf` file, it is time to run LILO to load the information into the MBR:

Code listing: Running LILO

```
# /sbin/lilo
```

LILO is configured and now your machine is ready to boot into Gentoo Linux!

Using framebuffer

People who have selected framebuffer in their kernel should add **vga=xxx** to their bootloader configuration file. **xxx** is one of the values in the following table:

	640x480	800x600	1024x768
8 bpp	769	771	773
16 bpp	785	788	791
32 bpp	786	789	792

LILO-users will have to add **vga=xxx** on top of their configuration file.

GRUB-users will have to append **vga=xxx** to the **kernel (hd0,0)...** line.

24. Creating Bootdisks

GRUB Bootdisks

Note: Don't forget to insert a floppy in your floppydrive before proceeding.

It is always a good idea to make a boot disk the first time you install any Linux distribution. This is a security blanket and generally not a bad thing to do. If your hardware doesn't let you install a working bootloader from the chrooted environment, you may **need** to make a GRUB boot disk. If you are in this camp, make a GRUB boot disk and when you reboot the first time you can install GRUB to the MBR. Make your bootdisks like this:

Code listing: Creating a GRUB Bootdisk

```
# cd /usr/share/grub/i386-pc/
# cat stage1 stage2 > /dev/fd0
```

Now reboot and load the floppy. At the floppy's **grub>** prompt, you can now execute the necessary **root** and **setup** commands.

LILO Bootdisks

Note: Don't forget to insert a floppy in your floppydrive before proceeding.

If you are using LILO, it is also a good idea to make a bootdisk:

Code listing: Making a Bootdisk

```
# dd if=/boot/your_kernel of=/dev/fd0
(This will only work if your kernel is smaller than 1.4MB)
```

25. Using GRP

GRP users can, at this point, install binary packages:

Code listing: Installing from GRP

```
# emerge -k xfree
```

CD 1 contains enough applications to install a working system with XFree86. Additionally, CD2 of the 2-CD GRP set contains other applications including KDE, GNOME, Mozilla and others. To install these packages, you will need to reboot into your new Gentoo system first (covered in the "Installation complete!" section near the end of this document). After you are running your basic Gentoo system from the hard drive, you can mount the second CD and copy files:

Code listing: Loading binary packages from CD2

```
# mount /dev/cdrom /mnt/cdrom
# cp -a /mnt/cdrom/packages/* /usr/portage/packages/
```

Now various other applications can be installed the same way. For example:

Code listing: Installing KDE from GRP

```
# emerge -k kde
```

26. Installation Complete!

Now, Gentoo Linux is installed. The only remaining step is to update necessary configuration files, exit the chrooted shell, safely unmount your partitions and reboot the system:

Note: **etc-update** can provide you with a list of configuration files that have newer versions at your disposal. Verify that none of the configuration files have a big impact (such as `/etc/fstab`, `/etc/make.conf`, `/etc/rc.conf`, ...). Merge the files that don't have such a big impact, remove the updates of the others or view the diff and manually update the configuration file.

Code listing: Rebooting the System

```
# etc-update
# exit
(This exits the chrooted shell; you can also type ^D)
# cd /
# umount /mnt/gentoo/boot
# umount /mnt/gentoo/proc
# umount /mnt/gentoo
# reboot
(Don't forget to remove the bootable CD)
```

Note: After rebooting, it is a good idea to run the **modules-update** command to create the `/etc/modules.conf` file. Instead of modifying this file directly, you should generally make changes to the files in `/etc/modules.d`.

If you have any questions or would like to get involved with Gentoo Linux evelopment, consider joining our gentoo-user and gentoo-dev mailing lists (more information on our mailing lists page). We also have a handy Desktop configuration guide that will help you to continue configuring your new Gentoo Linux system and a useful Portage user guide to help familiarize you with Portage basics. You can find the rest of the Gentoo Documentation here. If you have any other questions involving installation or anything for that matter, please check the Gentoo Linux FAQ. Enjoy and welcome to Gentoo Linux!

27. Gentoo-Stats

The Gentoo Linux usage statistics program was started as an attempt to give the developers a way to find out about their user base. It collects information about Gentoo Linux usage to help us in set priorities our development. Installing it is completely optional and it would be greatly appreciated if you decide to use it. Compiled statistics can be viewed at http://stats.gentoo.org/.

The gentoo-stats server will assign a unique ID to your system. This ID is used to make sure that each system is counted only once. The ID will not be used to individually identify your system, nor will it be matched against an IP address or other personal information. Every precaution has been taken to assure your privacy in the development of this system. The following are the things that we are monitoring right now through our "gentoo-stats" program:

installed packages and their version numbers CPU information: speed (MHz), vendor name, model name, CPU flags (like "mmx" or "3dnow") memory information (total available physical RAM, total available swap space) PCI cards and network controller chips the Gentoo Linux profile your machine is using (that is, where the `/etc/make.profile` link is pointing to).
We are aware that disclosure of sensitive information is a threat to most Gentoo Linux users (just as it is to the developers).

Unless you modify the gentoo-stats program, it will never transmit sensitive information such as your passwords, configuration data, shoe size... Transmission of your e-mail addresses is optional and turned off by default. The IP address your data transmission originates from will never be logged in such a way that we can identify you. There are no "IP address/system ID" pairs.
The installation is easy - just run the following commands:

Code listing: Installing gentoo-stats

```
# emerge gentoo-stats   (Installs gentoo-stats)
# gentoo-stats --new    (Obtains a new system ID)
```

The second command above will request a new system ID and enter it into `/etc/gentoo-stats/gentoo-stats.conf` automatically. You can view this file to see additional configuration options.

After that, the program should be run on a regular schedule (gentoo-stats does not have to be run as root). Add this line to your `crontab`:

Code listing: Updating gentoo-stats with cron

```
0 0 * * 0,4 /usr/sbin/gentoo-stats --update > /dev/null
```

The **`gentoo-stats`** program is a simple perl script which can be viewed with your favorite pager or editor: `/usr/sbin/gentoo-stats`.

28. Gentoo On Less-Common Hardware

Hardware ATA RAID

Users who want to install Gentoo on Hardware ATA RAID must pay attention to the next steps in order for them to succesfully install Gentoo Linux:

Be sure to start the LiveCD with the **`doataraid`** kerneloption. If you've forgotten to select **`doataraid`** during bootup, or the modules mysteriously didn't load, load them as needed:

Code listing: Loading RAID modules

```
# modprobe ataraid
For Promise Raid Controllers:
# modprobe pdcraid
For Highpoint Raid Controllers:
# modprobe hptraid
```

Some ATA RAID Controllers require you to reboot after partitioning; formatting will otherwise fail. Before chrooting, mount the devicetree into the new environment:

Code listing: Mounting /dev into /mnt/gentoo/dev

```
# mount -o bind /dev /mnt/gentoo/dev
```

During kernel configuration, select the required RAID options:

Code listing: RAID in the Linux Kernel Configuration

```
For Highpoint RAID controllers:
ATA/IDE/MFM/RLL support  --->
[*] HPT36X/37X chipset support
[*] Support for IDE Raid controllers
[*] Highpoint 370 software RAID
For Promise RAID controllers:
ATA/IDE/MFM/RLL support  --->
[*] PROMISE PDC202{46|62|65|67} support
and/or
[*] PROMISE PDC202{68|69|70|71|75|76|77} support
[*] Support for IDE Raid controllers
[*] Support Promise software RAID (Fasttrak(tm))
```

When using GRUB add **`--stage2=/boot/grub/stage2`** when running **`grub`** to the **`setup`** command:

Code listing: Installing GRUB for Hardware RAID systems

```
grub> root (hd0,0)
grub> setup --stage2=/boot/grub/stage2 (hd0)
grub> quit
```

Also, in the GRUB configuration be sure to point the **root** to the appropriate RAID device:

Code listing: grub.conf for RAID

```
title=My Gentoo Linux on RAID
root (hd0,0)
kernel (hd0,0)/boot/bzImage root=/dev/ataraid/dXpY
```

LILO users should set the **root** option to the appropriate RAID device:

Code listing: lilo.conf for RAID

```
image=/boot/bzImage
label=linux
read-only
root=/dev/ataraid/dXpY
```

If you still have problems installing Gentoo Linux on your Hardware RAID, be sure to report them on http://bugs.gentoo.org.

Thanks for using Gentoo Linux, and have fun with your new installation!