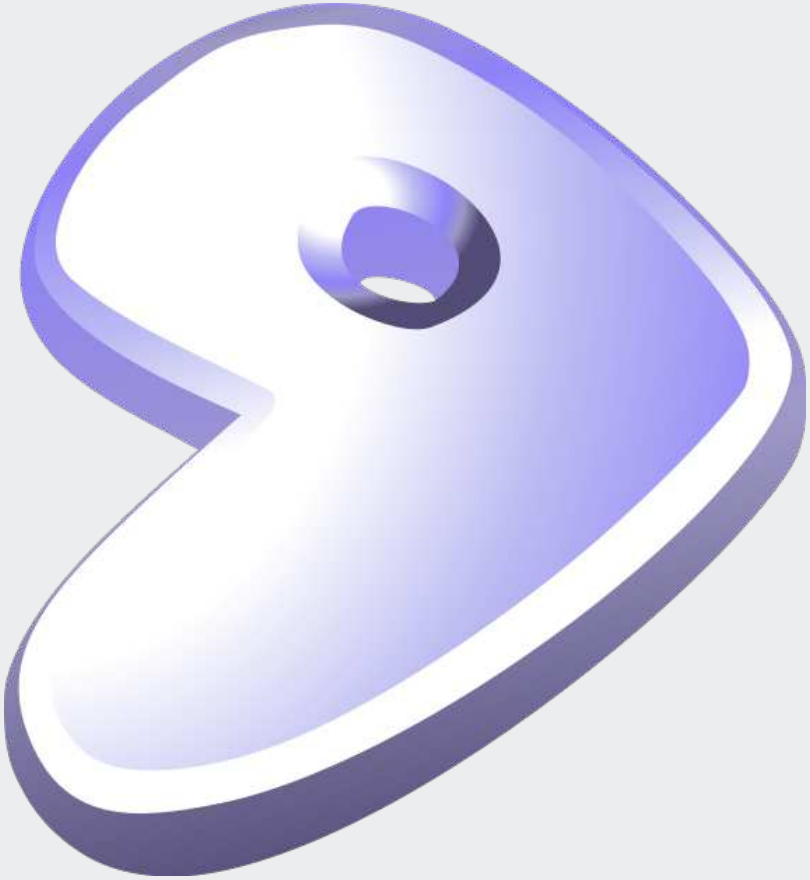
Gentoo Workshop



Mark Platek
Clarkson University
Fall 2010

Resources

- Gentoo Handbook:

  http://www.gentoo.org/doc/en/handbook/handbook-x86.xml

  http://www.gentoo.org/doc/en/handbook/handbook-amd64.xml

- Gentoo FAQs and HOWTOs:

  http://www.gentoo.org/doc/en/index.xml?catid=desktop

- Gentoo Forums

  http://forums.gentoo.org

- Myself – ask if you have questions!

  Email: platekme@clarkson.edu (or platekme89@gmail.com)

  IM: backslash54 v2

Now, where were we?

- When we left off last time, we had just finished compiling our kernels.
- There are still a few things left to do before we can boot.
- After everyone is able to boot their machines, it will be necessary to perform a full update. The goal of this session is to prepare you for that task.

Once again into the breech...

- We'll have to chroot. Last time, I promise!

  `mount /dev/sda3 /mnt/gentoo`
  `mount /dev/sda1 /mnt/gentoo/boot`
  `mkswap /dev/sda2`
  `swapon /dev/sda2`
  `mount -t proc none /mnt/gentoo/proc`
  `mount -o bind /dev /mnt/gentoo/dev`
  `chroot /mnt/gentoo /bin/bash`
  `env-update && source /etc/profile`
  `export PS1="(chroot) $PS1"`

Modules

- Some components of the kernel we compiled as modules. We can write a configuration file that tells the kernel to load these modules during the boot process.
- Invoke this command to see a list of modules printed:

  `find /lib/modules/2.6.34-gentoo-r12/ -type f -iname '*.o' -or -iname '*.ko'`
- The last things to be printed are the names of kernel modules that were compiled. In this case, we need to set the following modules to be loaded:

  test_nx
  scsi_wait_scan
  reiserfs
- Edit the file /etc/modules.autoload.d/kernel-2.6 and add each module's name on a single line (without the '.ko'). Now, once the initial boot process has completed, those modules will be loaded automatically.

fstab

- The file /etc/fstab defines which filesystems get mounted where at boot time.
- We'll have to add entries for all three partitions:

```
/dev/sda1  /boot   ext2    noauto,noatime 1 2
/dev/sda2  none    swap    sw             0 0
/dev/sda3  /       xfs     noatime        0 1
```

- And, for machines with more than 2G of RAM, we can do something special to speed up Portage: make the temporary space used during compilation (/var/tmp/portage) a ramdisk.

```
tmpfs   /var/tmp/portage   tmpfs   size=1500M,mode=0777   0 0
```

- Now, a 1.5G ramdisk will be created at boot time and mounted where portage leaves temporary files during compilation. This results in a marked improvement in compilation times.

Aside: rc-update

- rc-update is Gentoo's tool for managing init scripts.
- Init scripts live in /etc/init.d/ and usually start some kind of daemon.
- In Gentoo, each init script has dependencies. So, all rc-update needs to do is create a dependency tree to determine a safe order in which to start them all.
- Every time you add or remove a process from a given runlevel, the tree is recomputed.
- There are two main runlevels: default and boot. Only critical things should be added to the boot runlevel, everything else (mysql, apache, network, etc.) can go in default.
- Invoke `rc-update -s` to see all of the init script set to run at boot.

Networking setup

- Set a hostname! Edit the file /etc/conf.d/hostname.
- The file /etc/conf.d/net is where we shall define our network interfaces. For now, we'll just use DHCP. Add the line:

  ```
  config_eth0=( “dhcp” )
  ```

- Then use the rc-update tool to start networking at boot time:

  `rc-update add net.eth0 default`

- Which means, “add the 'net.eth0' init script to runlevel 'default'”.
- Don't forget to add your hostname to /etc/hosts! Use this syntax:

  ```
  127.0.0.1      hostname.localhost hostname localhost
  ```

Some more housekeeping...

- Set your root password: `passwd` will do the trick.
- Edit the file /etc/rc.conf and set your EDITOR variable. We'll be revisiting this file when it's time to install Gnome.
- If you have a non-us keyboard, you'll want to edit /etc/conf.d/keymaps.
- Edit /etc/conf.d/clock. Set CLOCK to “UTC” if your hardware clock is set to GMT, or set it to “local” if your hardware clock is set to local time. Also set TIMEZONE to “EST” so that /etc/localtime gets updated automatically.

Install system utilities

- First, install a system logger. There are several choices here, but we'll stick with syslog-ng and logrotate. Check out the Handbook for alternatives.

  `emerge syslog-ng logrotate`

- Finish by adding the logger's init script to the default runlevel:

  `rc-update add syslog-ng default`

- Now, install a cron daemon. The handbook recommends vixie-cron, but if you are used to crontab, dcron may suit you better.

  `emerge dcron && rc-update add dcron default`
  `crontab /etc/crontab`

- That last step is to set up the crontab at /etc/crontab.

System utilities (cont'd)

- You'll probably want to use locate, so `emerge slocate`.
- Now, let's install some utilities to support the filesystems we built into the kernel:

  `emerge xfsprogs reiserfsprogs`
- All ext filesystems are supported by default by the preinstalled package e2fsprogs. Re-emerge it if you don't want the binary version.
- Finally, install a DHCP client. Otherwise, net.eth0 will probably fail, and any init scripts depending on it will not be started.

  `emerge dhcp dhcpcd`
- You may also want to have gpm (a console-based mouse pointer) installed.

  `emerge gpm`

  `rc-update add gpm default`

Aside: grub syntax

- Grub has a funky syntax for addressing hard disks and partitions. It is:

  (hdx,y)

- ...where x is a number (starting at zero) that corresponds to the hard drive itself (so sda is 0, sdb is 1, sdc is 2, and so on)
- ...and y is another number (starting at zero) that corresponds to a partition on that hard drive (so the first partition is 0, the second partition is 1, and so on).
- So, for example, /dev/sda1 in grub syntax is (hd0,0). /dev/sda2 would be (hd0,1). /dev/sdb4 would be (hd1,3).

Bootloader: grub.conf

- Start by emerging grub: `emerge grub`. This will place a new configuration file: /boot/grub/grub.conf.
- Now, we'll write up our grub.conf. The important lines to include are:

```
title Gentoo 2.6.34-gentoo-r12
```

- ...which sets the title of the entry as you will see it when you boot up the machine

```
root (hd0,0)
```

- ...which sets /dev/sda1 as the partition where the kernel image can be found

```
kernel /boot/kernel-2.6.34-gentoo-r12 root=/dev/sda3 vga=ask
```

- ...which tells grub to boot from the kernel image at /boot/kernel-2.6.34-gentoo-r12 and specifies /dev/sda3 as the root partition
- But wait! What is this kernel image? Where did it come from? It was compiled when we built the kernel. It can be found at /usr/src/linux/arch/x86_64/boot/bzImage.
- Copy it to /boot, where grub expects it to be:

  `cp /usr/src/linux/arch/x86_64/boot/bzImage /boot/kernel-2.6.34-gentoo-r12`

- Finally, uncomment the splash image line to get a fancy-looking grub.

Bootloader (non-multiboot)

- Now, we'll install grub to the hard drive's MBR. Get to a grub command line:

  `grub --no-floppy`
- Tell grub where the /boot partition is:

  `root (hd0,0)`
- Tell grub where to install itself (to the hard disk's MBR):

  `setup (hd0)`
- And finish by leaving the grub command line:

  `quit`

Bootloader (multiboot)

- Things are a bit more complicated for multiboot systems...
- There are two options: using an existing grub installation, or chainloading a new grub installation to the /boot partition's root block.
- The fundamental problem is that when grub is installed to the MBR, it can only point to a single boot partition... Which is already in use by another OS!
- If we use the existing grub installation, we would have to copy our kernel images into the existing OS's /boot directory.
- If we chainload, we'll only need to make an entry in the existing grub.conf, no files need to be copied.
- We'll be using the chainloading method, since it is less disruptive to the existing OS, and allows us to operate the Gentoo installation identically to those of us who are not multibooting.

Bootloader (multiboot)

- The first step is the same – get to a grub command line:

  `grub --no-floppy`
- Tell grub where the boot partition is:

  `root (hd0,n)` where n represents the Gentoo /boot partition.
- Install grub *to the /boot partition*!

  `setup (hd0,n)`
- Exit the command line:

  `quit`
- Now, mount your existing Linux installation somewhere, and and a new boot entry to the grub.conf:

```
title Gentoo Workshop
root (hd0,n)
chainloader +1
```

The Moment of Truth

- Everything is prepared. It's time to boot into our new Gentoo systems!
- Exit the chroot and reboot the system.
- At this point, everyone should be able to boot their systems.
- We added 'vga=ask' as a kernel parameter, so we can specify a resolution for the vesa framebuffer to use.
- Remember the hex identifier you choose, so you can hardcode it into your grub.conf later on – replace 'ask' with the number (don't forget the 0x!).
- If you want to try uvesafb, check out the Framebuffer howto:

  http://en.gentoo-wiki.com/wiki/Framebuffer

Final Steps

- There are one or two more steps left before the install is finished.
- First, add a user:

  useradd -m -G users,wheel,audio -s /bin/bash <username>
- Then, add yourself to some groups:

  gpasswd -a <username> <group>
- (Protip: yourself to these groups: video, games, usb, portage, plugdev)
- Remember the stage3 and portage tarballs? They are in /, you can finally remove them :).

Portage configuration files

- So now we have a working Gentoo system. What do we do with it?
- There are several critical config files in /etc/portage:

  package.use

  package.keywords

  package.license
- We will examine each of these in context.

USE flags, revisited

- Remember the USE variable we commented out last session? Now is the time to re-enable it. Edit your make.conf to restore the USE variable.
- You can set per-package USE flags in /etc/portage/package.use.
- Syntax is:

  ```
  <package-category>/<package-name> <flag1> <flag2> <flag3> …
  ```

- You can use `equery` to find out which packages have which flags available. You'll have to install it, though: `emerge portage-utils gentoolkit`
- To set a flag, simply give its name. To deliberately unset it, prepend its name with a hyphen.
- For example, try to run `emerge -puDN world`. See how portage complains about a missing USE flag 'extras' for the 'sys-fs/udev' package? Fix it by adding the following to package.use:

  ```
  sys-fs/udev extras
  ```

- Now, if we run `emerge -puDN world` again, the USE flag is set.

Masked packages

- Some packages are deliberately held back in the portage tree. A newer version is available, but you can't install it by default.
- Packages are masked by different “keywords”, such as '~amd64' (which indicates that the masked package may not be stable on 64-bit systems).
- Truly beta-quality packages that could seriously break things are “hard-masked”. There is a way to unmask these, but we won't look at it.
- If you want to have the very latest and greatest software , you'll have to unmask some packages. Be prepared for problems, though!
- To unmask a package, add its full name to /etc/portage/package.keywords.
- For example, try to `emerge adobe-flash`. Notice that it's masked by the '~amd64' keyword. Let's unmask it by adding flash to /etc/portage/package.keywords:

  ```
  www-plugins/adobe-flash
  ```

- Now try to install it. It's still masked, but by a license!

Software licenses

- Some software, such as Sun's JDK/JVM and Adobe Flash are masked by a license that you have to agree to before you can install the software.
- Portage will tell you where to find the license text if you try to install a license-masked package.
- If you accept the license, add its name to /etc/portage/package.license like you would a USE flag:

  ```
  www-plugins/adobe-flash
  ```

- There is very little software in portage that must be added to package.license.

Next step: full update

- Our systems are working, but they are woefully out of date. The last time we synced the portage tree was weeks ago! So, sync with mirror: `emerge --sync`
- Run `emerge -puDN world` to see all the packages that need to be recompiled, and then `emerge -qvuDN to update. (We're using -qv here because scrolling text output is extra work that causes the update to take longer.)
- Look at all those packages! That's going to take a long time to compile. Guess what your homework is?

- Note: If you are up on your news bulletins, you will notice that system perl will be updated to an incompatible version when you update. Don't forget to run:

  `perl-cleaner --all`
- If you have problems, run `perl-cleaner --really-all`.

Cleaning up

- If updating a package causes things to break, the package maintainers will leave a message for us with information and/or instructions on how to fix the problem.
- We can see an example of this by emerging udev (remember, we added the use flag to it?)

  `emerge udev`
- After updating, it is critical that all of these messages are examined, and any instructions performed. It is easy to break things badly if you ignore these messages.
- Portage logs all messages to /var/log/portage/elog/summary.log. You can use that as a reference while cleaning up.

Next week

- What do you want to see for the last workshop session?
- Here are some ideas:

  X server installation/GNOME

  LAMP server

  ALSA sound configuration

  UTF-8 locale configuration

  Portage overlays

Copyright