Operating System Kernel Compilation

Sarah Diesburg
CS 3430
Operating Systems
Kernel Coding

- The next project involves compiling a new kernel!
  - Fun but challenging....

- Each person/team will get their own virtual machine(s)
- See website instructions for how to install/use the virtual machine
Project 2 Credentials

- Username: os
- Password: os

- Username: root
- Password: cmpt351
Basic System Commands

- $> passwd
  - Change your password

- $> su
  - Become root directly. Use root’s password when prompted.

- $> sudo
  - Do something with root’s privileges without having to become root
Basic System Commands

- `$> chown –R os:os <dir>`
  - Changes ownership of dir to user/group os
- `$> adduser <username>`
  - Adds user specified by username
- `$> userdel <username>`
  - Deletes user specified by username
Moving Files

- Must use Winscp or Filezilla
  - The way you communicated with the prog class machines
Moving Files
1) Start the VM, then open terminal
Moving Files

2) Get the VM’s IP address

```
os@debian:~$ /sbin/ifconfig
eth1  Link encap:Ethernet  HWaddr 08:00:27:4a:07:3d
       inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
       inet6 addr: fe80::a00:27ff:fe4a:73d/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:11 errors:0 dropped:0 overruns:0 frame:0
       TX packets:47 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:2185 (2.1 KiB) TX bytes:7491 (7.3 KiB)

eth2  Link encap:Ethernet  HWaddr 08:00:27:fc:bc:3b
       inet addr:192.168.56.101  Bcast:192.168.56.255  Mask:255.255.255.0
       inet6 addr: fe80::a00:27ff:febc:3b/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:58 errors:0 dropped:0 overruns:0 frame:0
       TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:9227 (9.0 KiB) TX bytes:11791 (11.5 KiB)

lo    Link encap:Local Loopback
       inet addr:127.0.0.1  Mask:255.0.0.0
```
Moving Files
3) Open WinSCP, type in info
Booting from Console

Kernel versions may be slightly different...

GNU GRUB  version 0.97  (638K lower / 2095040K upper memory)

Debian GNU/Linux, kernel 2.6.26-2-686
Debian GNU/Linux, kernel 2.6.26-2-686 (single-user mode)

Use the ↑ and ↓ keys to select which entry is highlighted. Press enter to boot the selected OS, 'e' to edit the commands before booting, or 'c' for a command-line.

The highlighted entry will be booted automatically in 4 seconds.
Booting

Start with this kernel – never modify this kernel!
Booting

This kernel is only if you really get into trouble…

GNU GRUB version 0.97 (638K lower / 2095040K upper memory)

Debian GNU/Linux, kernel 2.6.26-2-686
Debian GNU/Linux, kernel 2.6.26-2-686 (single-user mode)

Use the ↑ and ↓ keys to select which entry is highlighted. Press enter to boot the selected OS, ’e’ to edit the commands before booting, or ’c’ for a command-line.

The highlighted entry will be booted automatically in 4 seconds.
Kernel Compilation
Example of Kernel Compilation
Choosing a Kernel

- The next project will involve kernel development
  - We all need to be in the same boat...
- The class version will be 2.6.39.4
Kernel Repository

- [http://kernel.org](http://kernel.org) – place to download latest and greatest Linux kernels!
- Traditional place for kernel is in /usr/src/
- This has already been done for you
Compiling the Kernel

- Compiles based on configuration of `.config` file located in the kernel source directory

**How can I correctly configure my kernel?**

1. Use the `make menuconfig` command
2. Use the `make oldconfig` command
3. Combination of the two
‘make menuconfig’

- Manually select and deselect drivers and options for your kernel
  - * -- compiled directly into kernel
  - M – compiled into a module (driver) that can be dynamically added

- Can be tough to figure out what you need and don’t need
  - Command ‘lspci’ shows hardware on computer
‘make menuconfig’
`lspci`

```
user@cop4610:~$ lspci
00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host bridge (rev 01)
00:01.0 PCI bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX AGP bridge (rev 01)
00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA (rev 08)
00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
00:07.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:07.7 System peripheral: VMware Inc Virtual Machine Communication Interface (rev 10)
00:0f.0 VGA compatible controller: VMware Inc Abstract SVGA II Adapter
00:10.0 SCSI storage controller: LSI Logic / Symbios Logic 53c1030 PCI-X Fusion-MPT Dual Ultra320 SCSI (rev 01)
00:11.0 PCI bridge: VMware Inc Device 0790 (rev 02)
00:15.0 PCI bridge: VMware Inc Device 07a0 (rev 01)
02:00.0 USB Controller: Intel Corporation 82371AB/EB/MB PIIX4 USB
02:01.0 Ethernet controller: Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] (rev 10)
02:02.0 Multimedia audio controller: Ensoniq ES1371 [AudioPCI-97] (rev 02)
02:03.0 USB Controller: VMware Inc Abstract USB2 EHCI Controller
user@cop4610:~$ ```
General Linux Source Organization

- Memory Management
- Networking Layer
- Security
- Sound Processing

- Architecture-specific Code
- Disk Drivers
- Devices
- File Systems
‘make oldconfig’

- Build a new configuration file based on an old working configuration file!
  1. **Find** a working configuration file
     - Look in `/boot` for the current kernel’s config file
  2. **Copy** (don’t move) the current kernel’s config file into the new kernel source and rename it to ‘.config’
  3. **Issue** command ‘make oldconfig’ inside the new kernel source
  4. **Accept** new default changes
Combine the Methods

- Can use ‘make oldconfig’ to get a working config file
- Can use ‘make menuconfig’ to take some unneeded things out of the new kernel
  - Saves time when compiling
Config Example (Inside Source Dir)

$> cd /usr/src/linux-2.6.39.4

$> cp /boot/config-2.6.32-5-686 .config

$> make oldconfig

$> make menuconfig
Compiling the Kernel

- Issue ‘make’ inside the source directory

$> \text{make}$

- Grab some coffee....
Installing the Kernel

1. Install the modules (drivers)
   - $> sudo make modules_install

2. Install the actual kernel
   - $> sudo make install

What does ‘sudo’ do?
Final Step

- Create the ramfs image
- Why do we need an extra ramfs image?
Suppose you compiled the disk drivers and root file system drivers as modules

- Boot process loads the kernel image
- Kernel image does not contain disk drivers
Boot Driver Solutions

- **Hard**: Figure out all drivers needed to read the root file system and compile into kernel
  - May take a few tries…

- **Easy**: Create a root file system image file that will load the right modules for you
  - initramfs
Create the initramfs image

$> cd /boot

$> sudo mkninitramfs -o /boot/initrd.img-2.6.39.4
2.6.39.4
Configure the Boot Loader (Grub)

$> sudo update-grub

- Now you can issue a reboot and cross your fingers!

$> sudo reboot
Telling Which Kernel You are In

- Select the new kernel from the grub menu!
- Watch your virtual machine
  - If something goes wrong, reboot and go back into your original kernel to try again
- The following command will tell you which kernel you are in:

  $> \texttt{uname -r}$
Uh oh....

- If it doesn’t boot, go back into the original kernel
  - Always important to have one working kernel!
- Make sure you didn’t skip a step...
- You may have taken too much out of the kernel...
  - Start with bloated kernel, take things out one by one
- Look to additional resources...
Additional Resources

- http://www.tuxradar.com/content/how-compile-linux-kernel
Next Recitation

- More on Project 2
  - Loading a kernel module and playing with proc files