

Recitation Week #6

Compiling the Kernel

Alejandro Cabrera
Operating Systems
COP4610 / CGS5765

Today's Recitation

- Syllabus Updates
- Project 2 Specification Updates
- Introducing the Kernel
- Obtaining the Kernel Source
- Configuring the Kernel
- Compiling the Kernel
- Installing the Kernel
- Booting Into the New Kernel
- The Organization of the Kernel

Syllabus Updates

- Removed legacy requirements
 - No longer need you to use your SSN anywhere!
- Clarified project submission procedure
 - No executables
 - p#-opsys-<last_name1>-<last_name2>.tar.gz
 - Text file README and final project report.

Project 2 Specification Updates

- Clarified I/O redirection
 - $\text{CMD} < \text{FILE}$
 - $\text{CMD} > \text{FILE}$
 - $\text{CMD}_1 | \text{CMD}_2$
- Corrected error in piping specification
 - $\text{CMD}_1 | \text{CMD}_2 | \text{CMD}_3$
 - Means execute CMD_1 then pipe to CMD_2 then pipe to CMD_3

The Linux Kernel

- Welcome!
- The Linux kernel is composed of over 10 million lines of code that:
 - Control your CPU (scheduler)
 - Manage your memory (memory manager, page tables)
 - Organize your files on storage (file system)
 - Operate your video card (device drivers)
 - Allow you to connect to the internet (networking)
 - Protect your computer from attacks (security)
 - More!

Kernel Preparation Timetables

- Let this be a guide as to how long the process will take:
 - Obtain kernel: 5 – 15 minutes
 - Unpack kernel: 1 – 3 minutes
 - Investigate machine: 1 – 5 minutes
 - Configure kernel: 30 – 90 minutes
 - Build kernel: 45 – 120 minutes
 - Install kernel: 1 – 5 minutes
 - Boot kernel: 5 – 10 minutes
- Total: 88 – 248 minutes!

Obtaining the Kernel

- www.kernel.org/
- We'll be using 2.6.31.1 for this course
 - 2.6.31.1 is the latest stable version of the kernel
- The bleeding edge is 2.6.32-rc1!

Unpacking the Kernel

- Find your downloaded kernel file
- Move the source to a directory you have access permissions to, e.g. /usr/local/src.
- Execute the following:

```
$> tar jxvf linux-2.6.31.1.tar.bz  
$> cd linux-2.6.31.1  
$linux-2.6.31.1>
```

Kernel Configuration

- The kernel configuration step is the trickiest.
- Here, you're going to tell the kernel all about your machine.
- You'll need to know your hardware.
- Take great care in this step.
 - If you overlook information here, you may end up with an inadequate kernel to boot your machine
 - This means you'll have to re-configure and rebuild your kernel (75 – 210 MORE minutes)
- On the following slide, we'll introduce a few commands to perform investigation with.

Machine Investigation

```
# Queries the bus for hardware info
```

```
$> lspci > hardware.log
```

```
# Now for the CPU(s)
```

```
$> cat /proc/cpuinfo > cpu.log
```

Machine Investigation Files

- Keep hardware.log and cpu.log handy.
- You'll need these files for the configuration process.

Kernel Configuration

- Perform the following commands:

```
# Cleans up artifacts from previous
```

```
# build
```

```
$linux-2.6.31.1> make mrproper
```

```
# Builds configuration system
```

```
$linux-2.6.31.1> make menuconfig
```

menuconfig: A Visual

- Should look like this:

```
.config - Linux kernel v2.6.30.5 Configuration

Linux Kernel Configuration

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M>
modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

General setup --->
[*] Enable loadable module support --->
-*- Enable the block layer --->
Processor type and features --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Executable file formats / Emulations --->
-*- Networking support --->
Device Drivers --->
Firmware Drivers --->
File systems --->
Kernel hacking --->
Security options --->
-*- Cryptographic API --->
[*] Virtualization --->
Library routines --->
---
Load an Alternate Configuration File
Save an Alternate Configuration File

<Select> < Exit > < Help >
```

Setting Parameters

- To build a kernel that'll run, at a minimum, you must be sure that:
 - The right CPU type is set
 - Processor types and features → Processor family
 - The right device drivers are enabled
 - Device Drivers → *
 - The right bus parameters are set:
 - Bus options → *
 - The right file system is set:
 - File systems → *

Configuration Advice

- You are not trying to build a *minimal* kernel, you are trying to build a *boot-able* kernel:
 - If you are unsure you need a parameter set, set it, just to be sure.
- A minimal kernel might build and boot faster, but if you missed something...
- **SAVE** your configuration file and back it up.
 - menuconfig has an option to help you save your configuration.
- In the worst case, you could enable:
 - ALL file systems
 - ALL device drivers
 - ALL bus options
- You still have to get your CPU correct...

Compiling the Kernel

- With the configuration complete, issue the following command:

```
$linux-2.6.31.1> make
```

- Wait until the first 10 or so lines display.
- If all goes well, go out to lunch, go read a book, go take a nap, go work on project 2.
 - It'll be awhile before the kernel finishes compiling.

Installing the Kernel

- With the build process complete, you'll now need to install the kernel.
- This is a straightforward process, though the commands are likely new to you, unless you've installed a kernel in the past.

Installing the Kernel: Commands

```
$linux-2.6.31.1> su  
<enter your password>  
#linux-2.6.31.1> make install  
#linux-2.6.31.1> cd /boot  
#boot> mkinitramfs -o initrd-  
    2.6.31.1.img 2.6.31.1  
#boot> update-grub
```

Installing the Kernel: Alternate Commands

```
$linux-2.6.31.1> su  
<enter your password>  
#linux-2.6.31.1> make install  
#linux-2.6.31.1> cd /boot  
#boot> mkinitramfs -o initrd-  
    2.6.31.1.img 2.6.31.1  
#boot> sudo nano grub/menu.lst  
# make an entry for the new kernel  
# as explained next.
```

Grub

- Grub is the boot loader.
- Handles booting into your kernel.
- `grub/menu.lst` has entries for every kernel installed on your machine.
- You'll need to modify `menu.lst` if the command *update-grub* is not available on your distribution.
- Use the existing `menu.lst` as reference.

Editing grub/menu.lst

- title – Put whatever you want here. It's just the name that appears on the boot screen.
- kernel – The kernel boot image path.
- initrd – The kernel image path.
- All else – boot options

```
title <Your Descriptive Kernel Name>
kernel /boot/vmlinuz-2.6.31.1
initrd /boot/initrd.image-2.6.31.1
    root=<copy root from other entry>
boot
```

Booting Into the Kernel

- All that's left to do is to restart the machine.
- When the boot loader comes up, choose your kernel.
- Assuming all went well, you should be watching the machine boot up.
- If you made it to the log in screen, congratulations!
- Else, go back to the kernel configuration step.

Next Time:

- Kernel Source Structure
- Kernel Modules
- Compiling Kernel Modules
- Module commands
 - insmod
 - rmmod
 - lsmod
 - modprobe
- Project 3

Any Questions?