TNE30019/TNE80014 – Unix for Telecommunications

Compiling and Installing the Kernel

Dr. Jason But

Swinburne University

Outline

Why Recompile Kernel?

- Distributions come with pre-compiled kernel
- Many users don't bother, but there are some good reasons
- Because it's cool
- Because you need smallest possible kernel for embedded device
- Because you need to modify kernel config (e.g. HZ=1000)
- Because somebody coded useful patch or new driver not (yet) part of official kernel distribution

Recompiling the Kernel

- FreeBSD kernel source code lives in `/usr/src/sys`
- Linux kernel source code lives in `/usr/src/linux`
- Kernel is written in C (and CPU-specific assembler)

Step-by-step process

- Configure kernel
- Compile kernel and modules
- Install kernel
- Update boot manager (optional)

Compilation process

- C source files are compiled to object files using C compiler (**gcc** – GNU C Compiler)
- Object files are linked into one executable using linker (**ld**)
  - Main kernel
  - Module objects
- How do we manage compilation of tens of thousands of files without missing one?
- How do we make small code change and rebuild kernel without compiling everything?

make and Compiling

- We need some automated means of compiling kernel

make

- Tool that uses project configuration file(s) (`Makefile`)
- Specifies which files need to be compiled prior to linking
- Checks object files against source files and only compiles source **IF** it is newer (changed since last compile)
- Manages linking

- Kernel is built by typing "`make`"

FreeBSD Configuring and Building

- Kernel configurations in `/usr/src/sys/<platform>/conf`
- For 64-bit PCs `<platform>` is `amd64`
- Configuration file name: `<config_name>`

Compiling

```
cd /usr/src
make buildkernel KERNCONF=<config_name>
```

Installing

```
cd /usr/src
make installkernel KERNCONF=<config_name>
```

- This will move the old directory to `/boot/kernel.old`
- Copy newly compiled kernel to `/boot/kernel`

Linux Configuring and Building

- Sometimes distribution-specific steps
- Here generic steps for vanilla kernel (www.kernel.org)

Preparing

`cd /usr/src/linux`
`make mrproper` (only if need cleanup from previous compile)
`make oldconfig` or `make menuconfig`

Compiling

```
make
```

Linux Installing

- Kernel lives in `/boot/vmlinuz-<kernel-version>`
- Modules live under `/lib/modules/<kernel-version>/`

Installing

```
make modules_install
make install
```

- This will put new kernel into `/boot`
- This will put new modules into `/lib/modules/`

Create initial RAM disk (if make install doesn't)

```
cd /boot
mkinitrd -o initrd.img-<kernel-version>
<kernel-version>
```


Rebooting

- Current kernel will still be running
- On FreeBSD/Linux, when program runs, its entire code is in memory – you can delete program but it will still run

Running your new kernel

- Update boot manager if needed (depends on boot manager)
- Reboot system
  `shutdown -r now` or `reboot`
- New kernel will automatically load

What if you "Stuff it Up" *TM*

Source: http://www.quickmeme.com/meme/3qmy44

What if you "Stuff it Up" *TM*

- Your new kernel doesn't work
- System won't boot
- You need to recover old kernel

Recovery Process

1. Boot recovery system
   - Good kernel on hard disk selectable in boot manager (if exists)
   - Good kernel from CD/DVD image
2. Mount local hard disk so you have access to files
   - FreeBSD: rename `/boot/kernel.old` to `/boot/kernel`
   - Linux: relink vmlinuz and initrd to good kernel and good initrd and update boot manager if necessary
3. Reboot