

Linux Kernel Building

The formula for letting Linux remanufacture itself
(a.k.a. “reconfiguring the kernel”)

David Morgan

Linux Kernel

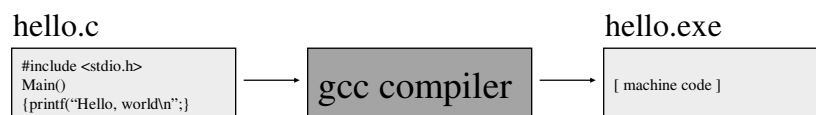
- Is the machine code of the operating system
- Runs from within memory
- Gets there by loading in from a file
- File made by compilation from source code
- The source code is on your hard disk

Compilation

- Translates source code to executable code
- Usually operates in several steps
- Steps may each be performed manually
- Or automated and consolidated with “make” utility

Manual Compilation

```
[root@EMACH1 root]# ls
hello.c
[root@EMACH1 root]# gcc hello.c -o hello.exe
[root@EMACH1 root]# ls
hello.c hello.exe
[root@EMACH1 root]# ./hello.exe
Hello, world
[root@EMACH1 root]#
```



Automated Compilation

- Uses the “make” utility
- With its script of instructions “Makefile”
- User writes Makefile then runs make
- Special kind of conditional script/batch

Make utility vs. shell script

- **ordinary shell script**
 - Invocation: type script file’s name
 - Conditionality: script language “if” construct
- **make**
 - Invocation: type “make” (reads Makefile)
 - Conditionality: file timestamp relationships

Operation of make

Contents of Makefile:

```
hello: hello.c
    gcc hello.c -o hello
    gcc hellodolly.c -o hellodolly
    gcc wellhello.c -o wellhello
```

When “make” runs, the block of commands may be executed, according to whether the target file is older than the file(s) it depends on.

Make can have multiple targets

Contents of Makefile:

```
hello: hello.c
    gcc hello.c -o hello
    gcc hellodolly.c -o hellodolly
    gcc wellhello.c -o wellhello

goodby: goodby.c
    gcc goodby.c -o goodby
```

make hello – triggers evaluation of upper block

make goodby – triggers evaluation of lower block

Linux – a Makefile compiles it

- Located in /usr/src/linux
 - a symbolic link e.g. linux -> linux-2.6.9
- Multiple invocation keywords/targets, e.g.
 - make clean
 - make modules_install
- Source code lies in /usr/src/linux subtree
- Instructions: /usr/src/linux/README

Kernel.org to get patches or full tree

The Linux Kernel Archives

Welcome to the Linux Kernel Archives. This is the primary site for the Linux kernel source, but it has much more than just kernels.

Protocol	Location
HTTP	http://www.kernel.org/pub/
FTP	ftp://ftp.kernel.org/pub/
RSYNC	rsync://rsync.kernel.org/pub/

The latest stable version of the Linux kernel is: [2.6.9](#) 2004-10-18 22:10 UTC [F](#) [V](#) [VI](#) [C](#) [Changelog](#)

The latest [prepatch](#) for the stable Linux kernel tree is: [2.6.10-rc1](#) 2004-10-22 22:12 UTC [V](#) [C](#) [Changelog](#)

The latest [snapshot](#) for the stable Linux kernel tree is: [2.6.10-rc1-bk6](#) 2004-10-27 11:33 UTC [V](#)

The latest 2.4 version of the Linux kernel is: [2.4.27](#) 2004-08-07 23:28 UTC [F](#) [V](#) [VI](#) [C](#) [Changelog](#)

The latest [prepatch](#) for the 2.4 Linux kernel tree is: [2.4.28-rc1](#) 2004-10-22 21:22 UTC [V](#) [VI](#) [C](#) [Changelog](#)

The latest [snapshot](#) for the 2.4 Linux kernel tree is: [2.4.28-rc1-bk1](#) 2004-10-27 09:48 UTC [V](#)

The latest 2.2 version of the Linux kernel is: [2.2.26](#) 2004-02-25 00:28 UTC [F](#) [V](#) [C](#) [Changelog](#)

The latest [prepatch](#) for the 2.2 Linux kernel tree is: [2.2.27-pre2](#) 2004-04-20 19:26 UTC [V](#) [VI](#) [C](#) [Changelog](#)

The latest 2.0 version of the Linux kernel is: [2.0.40](#) 2004-02-08 07:13 UTC [F](#) [V](#) [VI](#) [C](#) [Changelog](#)

The latest [-ac](#) patch to the stable Linux kernels is: [2.6.9-ac4](#) 2004-10-24 18:59 UTC [V](#)

A compilation recipe for Linux

- `cd /usr/src/linux`
- `make menuconfig`
 - answer the questions (result: `/usr/src/linux/.config`)
 - see Help within menuconfig for explanation of questions
- `make`
- `make modules_install`

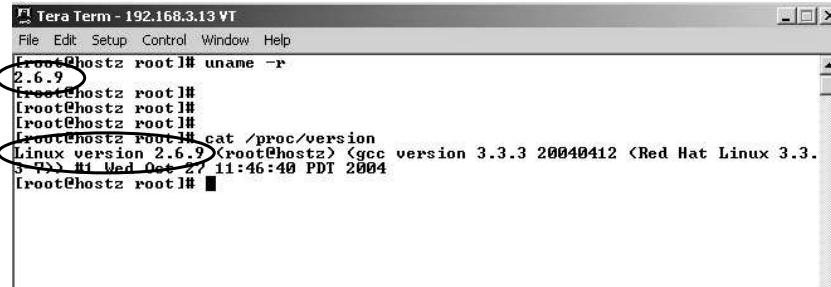
Using the new kernel

- Deposited in `/usr/src/linux/arch/i386/boot/bzImage`
- Copy it to `/boot` under a name that identifies it
e.g., `vmlinuz-2.6.9`
- Add a `/boot/grub/grub.conf` stanza for a new boottime option, e.g.

```
title My Newborn Linux (2.6.9)
  root (hd0,0)
  kernel /vmlinuz-2.6.9 root=/dev/hda2
```
- `mkinitrd /boot/initrd-2.6.9.img 2.6.9` (optional)
 - plus “`initrd /initrd-2.6.9.img`” in `grub.conf`
- Keep your working default kernel till the new one proves out!

Verifying what kernel is running

- `uname -r`
- `cat /proc/version`



```
Tera Term - 192.168.3.13 VT
File Edit Setup Control Window Help
[root@hostz root]# uname -r
2.6.9
[root@hostz root]#
[root@hostz root]#
[root@hostz root]#
[root@hostz root]# cat /proc/version
Linux version 2.6.9 (root@hostz) (gcc version 3.3.3 20040412 <Red Hat Linux 3.3.
5-7>) #1 Wed Oct 27 11:46:40 PDT 2004
[root@hostz root]#
```

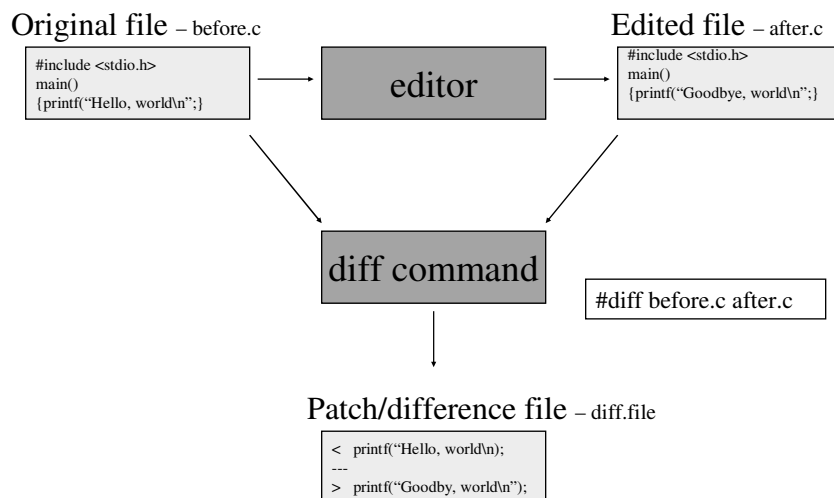
Why recompile Linux?

- To change configuration options (i.e., to incorporate a different set of components)
- To upgrade existing components by patching
- To move components to outboard modules (dll's) or bring modules into the kernel
- To upgrade to a new version
 - e.g., 2.2.12 to 2.2.14
 - install new source tree from www.kernel.org
 - change `linux` -> `linux-2.2.14`

Where to Get More Information

- /usr/src/linux/README
- <http://www.digitalhermit.com/linux/Kernel-Build-HOWTO.html>
- <http://www.rhil.net/docs/kernelbuild-26.html>

Frequent precursor: source code patching



Patching: getting the edited file without editing

Original file – before.c

```
#include <stdio.h>
main()
{printf("Hello, world\n");}
```

Patch file – diff.file

```
< printf("Hello, world\n");
---
> printf("Goodbye, world\n");
```

patch command

```
#patch before.c diff.file
```

Edited file – after.c

```
#include <stdio.h>
main()
{printf("Goodbye, world\n");}
```