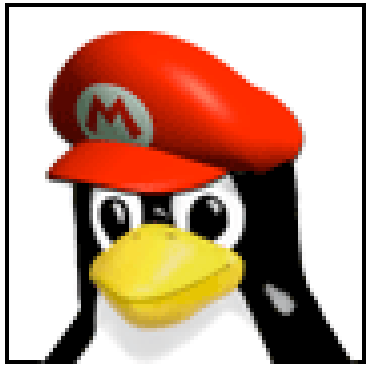
Linux Everywhere

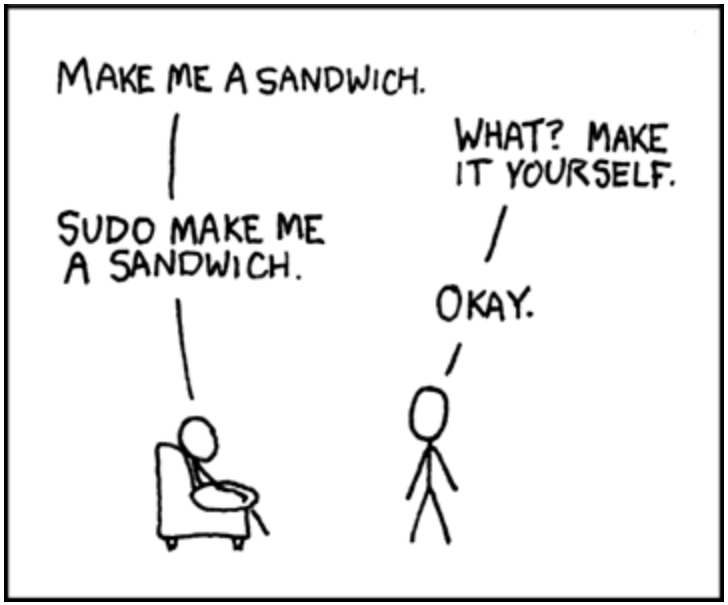
A look at Linux outside the world of desktops

CIS 191 Spring 2012 – Guest Lecture by Philip Peng

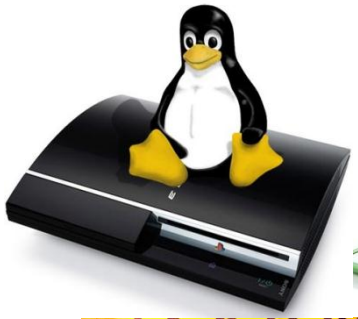


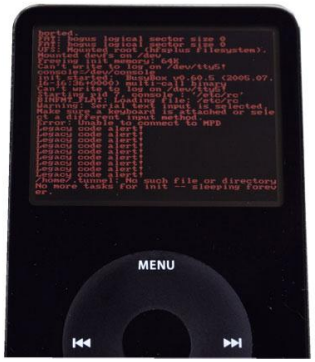
Lecture Outline

1. Introduction
2. Different Platforms
3. Reasons for Linux
4. Cross-compiling
5. Case Study: iPodLinux
6. Questions



What's in common?





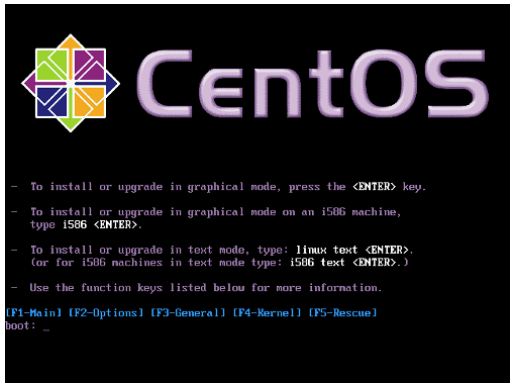
All your hardware are belong to us

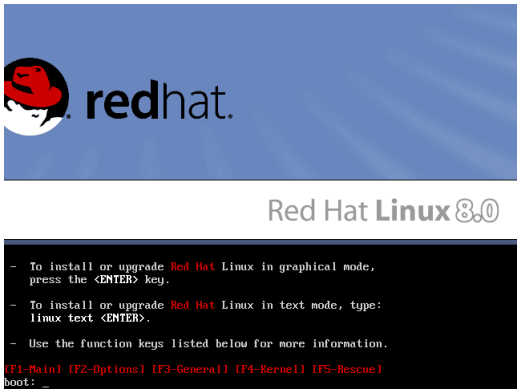
- Linux is everywhere
  - If its programmable, you can put Linux on it!
  - Yes, even a microwave

CES 2010, microwave running Android: http://www.handlewithlinux.com/linux-washing-cooking

Servers

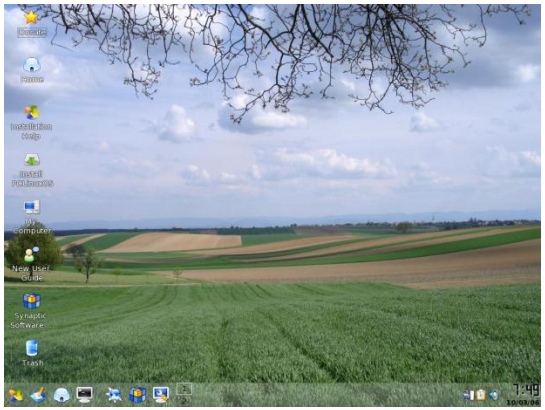
- What servers use
  - Stability, security, free
  - Examples:
    - CentOS
    - Debian
    - Red Hat





Desktop

- What you use
  - Free Windows/Mac alternative
  - Examples:
    - Ubuntu
    - Fedora
    - PCLinuxOS



Gaming Devices

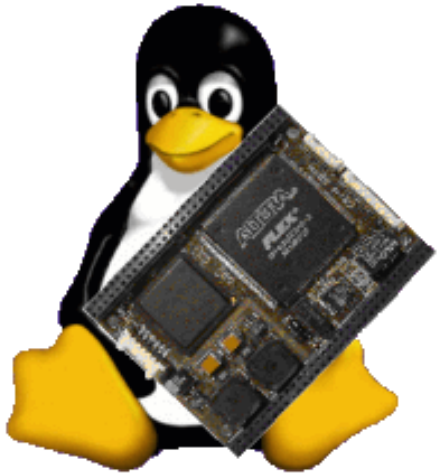
- What (white-hat) hackers do
  - To run “homebrew” software
  - Examples:
    - PS3, Wii, XBOX
    - PS2, GameCube
    - Dreamcast
    - PSP, DS
    - Open Pandora, GP2X

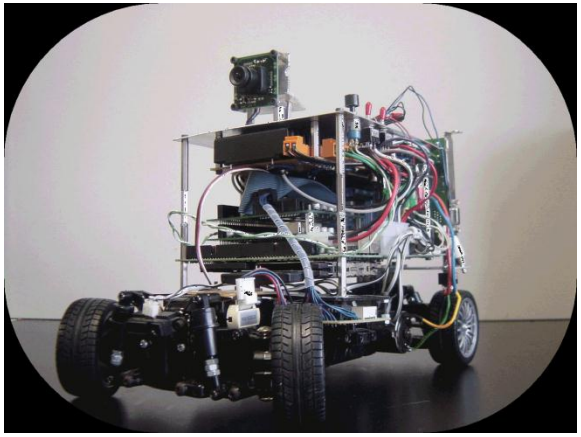
Mobile Devices

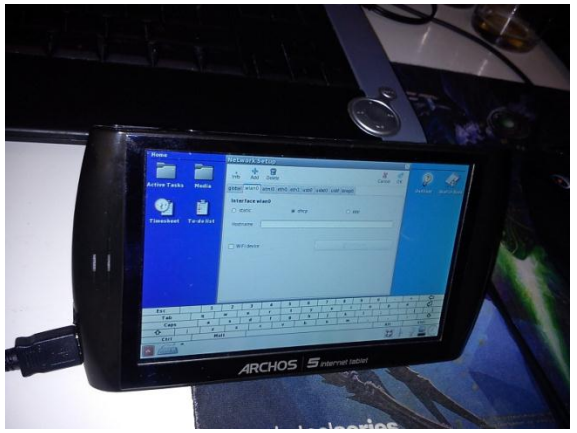
- What distributors are developing
  - Community contribution
  - Examples
    - Android
    - Maemo/MeeGo/Tizen
    - Openmoko

Embedded Devices

- What embedded hardware run
  - Small footprint, dev tools
  - Examples
    - RTLinux (real-time)
    - µClinux (no MMU)
    - Ångström (everything)







Why?


Linux
Windows
OS X
From the point of view of a...
Mac fan
Win Fan
Linux Fan
YOU
Microsoft

Free!

- Free!
  - As in freedom, i.e. open source
  - As in beer, i.e. vs paid upgrades

Homebrew!

- Run own software
  - Your hardware → your software?

Support!

- Community contribution
  - “For the greater good” (i.e. users)
  - Everyone contributes
    - Specialists from all over the world
  - Existing hardware support
    - Many already supported computer architecture
    - Modify existing drivers

Lots of support!

List of Linux supported architectures

From Wikipedia, the free encyclopedia

The **Linux kernel** is portable and supports the following computer architectu

- Alpha architecture:
  - DEC Alpha
  - Samsung Alpha CPU
- Analog Devices
  - Blackfin (since 2.6.22)
- ARM architecture:
  - Acorn Archimedes and Risc PC series
  - DEC StrongARM
  - Marvell (formerly Intel) XScale
  - Sharp Zaurus
  - iPAQ
  - Palm, Inc.'s Tungsten Handheld[1]
  - Gamepark Holdings' GP2X
  - Open Pandora
  - Nokia 770 Internet Tablet
  - Nokia N800
  - Nokia N810
  - Nokia N900
  - gumstix
  - Nintendo DS via DSlinux
  - Sony Mylo
  - Psion 5, 5MX, Series 7, netBook
  - Some Models of Apple iPods (see iPodLinux)
  - OpenMoko Neo 1973
  - Freescale's (formerly Motorola's) i.MX multimedia processors
- Atmel AVR32
- Axis Communications' ETRAX CRIS
- C6X from Texas Instruments
- Freescale's (formerly Motorola's) 68k architecture (68020, 68030, 68040,
  - Some Amigas: A1200, A2500, A3000, A4000
  - Apple Macintosh II, LC, Quadra, Centris and early Performa series
- Fujitsu FR-V
- Hexagon from Qualcomm
- Hewlett-Packard's PA-RISC family
- H8 architecture from Renesas Technology, formerly Hitachi.
  - H8/300
  - H8/500
- IBM
  - System/390 (31-bit)
  - Z/Architecture (Z mainframes) (64-bit)
- Intel IA-64 Itanium, Itanium II
- x86 architecture:
  - IBM PC compatibles using IA-32 and x86-64 processors:
    - Intel 80386, 80486, and their AMD, Cyrix, Texas Instruments and IBM varia
    - The entire Pentium series and its Celeron and Xeon variants
    - The Intel Core processors
    - AMD 5x86, K5, K6, Athlon (all 32-bit versions), Duron, Sempron
    - x86-64: 64-bit processor architecture, now officially known as AMD64 (A
    - Cyrix 5x86, 6x86 (M1), 6x86MX and MediaGX (National/AMD Geode) serie
    - VIA Technologies Eden (Samuel II), VIA C3, and VIA C7 processors
  - Microsoft's Xbox (Pentium III processor), through the Xbox Linux project
  - SGI Visual Workstation (Pentium II/III processor(s) with SGI chipset)
  - Sun Microsystems Sun386i workstation (80386 and 80486)
  - Support for 8086, 8088, 80186, 80188 and 80286 CPUs is under development
- M32R from Mitsubishi
- Microblaze from Xilinx
- MIPS architecture:
  - Dingoo
  - Infineon's Amazon & Danube Network Processors
  - Ingenic Jz4740
  - Jazz
  - Cobalt Qube, Cobalt RaQ
  - DECstation
  - Loongson (MIPS-compatible), Loongson 2, and Loongson 2E from BLX IC Desi
  - Some PlayStation 2 models, through the PS2 Linux project
  - PlayStation Portable uClinux 2.4.19 port [1]
  - Broadcom wireless chipsets
  - Dreambox (HD models) [3]
  - Cavium Octeon packet processors
- MN103 from Panasonic Corporation
- OpenRISC
  - OpenRISC 1000 family in the mainline Linux Kernel as of 3.1.
  - Beyond Semiconductor OR1200
  - Beyond Semiconductor OR1210
- Power Architecture:
  - IBM Servers
- PowerPC architecture:
  - IBM's Cell
  - Most pre-Intel Apple computers (all PCI-based Power Macintoshes, limited s
  - Clones of the PCI Power Mac marketed by Power Computing, UMAX and Mc
  - Amigas upgraded with a "Power-UP" card (such as the Blizzard or CyberS
  - AmigaOne motherboard from Eyetech Group Ltd (UK)
  - Samantha from Soft3 (Italy)
  - IBM RS/6000, iSeries and pSeries systems
  - Pegasos I and II boards from Genesi
  - Nintendo GameCube and Wii, through Nintendo GameCube Linux
  - Project BlackDog from Realm Systems, Inc.
  - Sony PlayStation 3
  - Microsoft's Xbox 360, through the free60 project
  - V-Dragon CPU from Culturecom.
  - Virtex II Pro Field Programmable Array (FPGA) from Xilinx with PowerPC co
  - Dreambox (non-HD models) [4]
- SPARC
  - SPARC (32-bit):
    - Sun-4 (to be abandoned in version 2.6.27)
    - SPARCstation/SPARCserver series (sun4c, sun4m, sun4d)
    - LEON
  - UltraSPARC (64-bit):
    - Sun Ultra series
    - Sun Blade
    - Sun Fire
    - SPARC Enterprise systems, also the based on the UltraSPARC T1, Ultra
- SuperH
  - Sega Dreamcast (SuperH SH4)
  - HP Jornada 680 through Jlime distribution (SuperH SH3)
- S+core
- Tilera
- Xtensa from Tensilica

Why not?

- Because we can
  - If its hackable, it can run Linux

How?

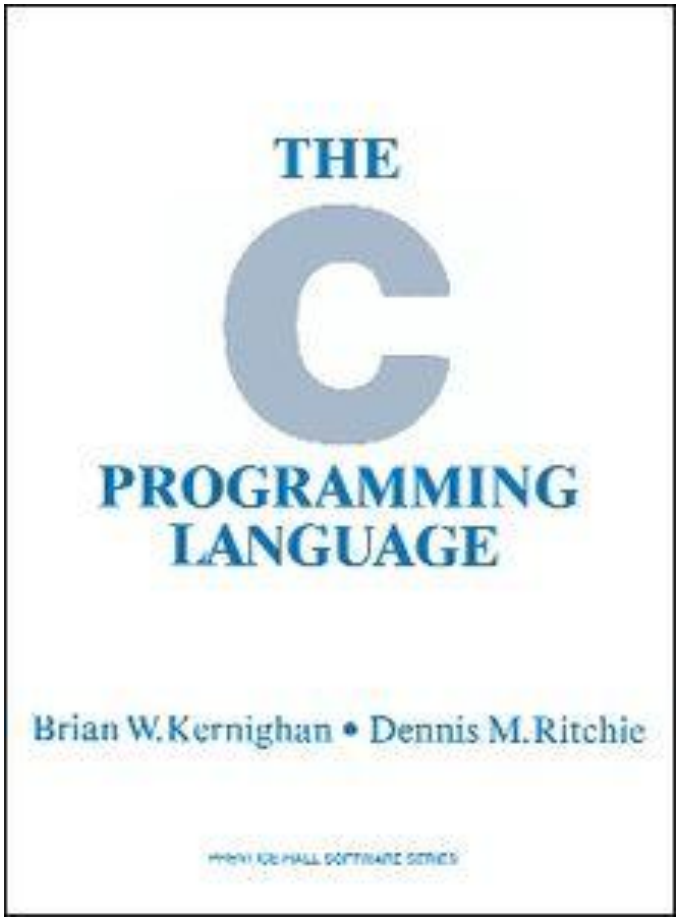
- How do we get Linux running on XXX?
- **Port:** A version of software modified to run on a different target platform
  - The PS3 *port* of Fedora is a modified build of Fedora compiled to run on the PS3 architecture
  - e.g. “I *ported* the Linux kernel to my iPod”

Cross-compiling!

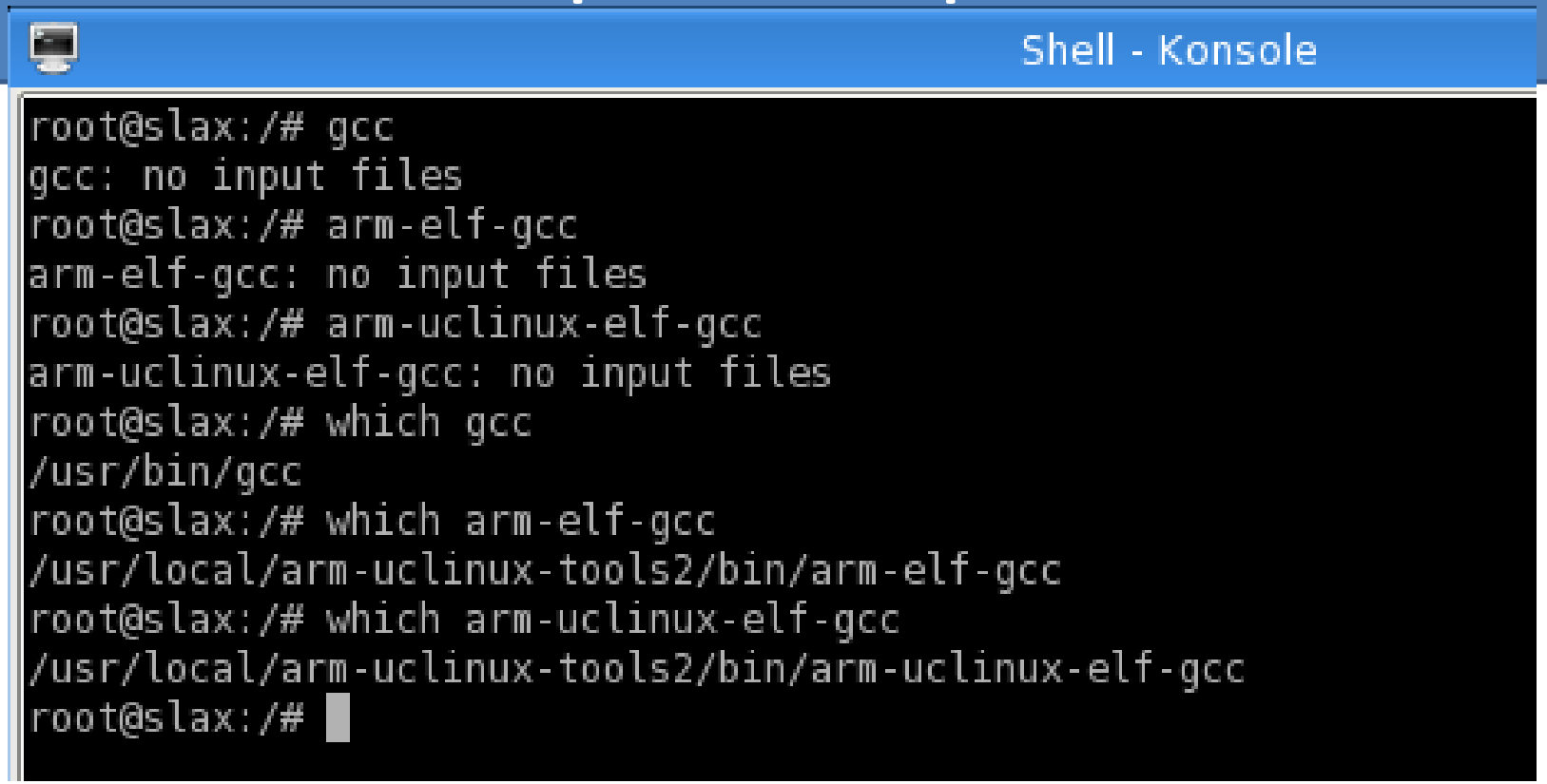
- Supported hardware? Easy! Cross-compile!
- **Compiler**: A program that converts code to an executable program for *your computer*
- **Cross-compiler**: A program that converts code to an executable program for *another platform*

Cross-compiling!

- What makes this possible? C and gcc
  - C programming language is made to be easily portable to different architectures
  - The Linux kernel and all basic tools are written in C
  - Same source code runs of all sorts of platforms



Multiple Compilers



- "`arm-elf`" is the architecture that runs `ELF` executables (default format for Linux) on an ARM processor
- µClinux is a Linux kernel fork for microcontrollers without a MMU (memory management unit)

Compiling for LFS (i368 Linux)

- Compiling `tar` for `LFS` (you did this for HW!)

```
# wget
http://ftp.gnu.org/gnu/tar/tar-
1.26.tar.bz2
# tar -xf tar-1.26.tar.bz2
# cd tar-1.26
# ./configure
# make
```

Compiling for arm-elf

- Compiling `tar` for `arm-elf` architecture

```
# wget
http://ftp.gnu.org/gnu/tar/tar-
1.26.tar.bz2
# tar -xf tar-1.26.tar.bz2
# cd tar-1.26
# ./configure CC=arm-elf-gcc
LDFLAGS=-elf2flt --host=arm-elf
# make
```

Compiling for arm-elf

- `# ./configure` **`CC=arm-elf-gcc LDFLAGS=-elf2flt --host=arm-elf`**
- `CC=arm-elf-gcc`
  - Specify the “cross-compiler” to be used
- `LDFLAGS=-elf2flt`
  - Set any special linking flags (e.g. target specific)
  - In this case, convert ELF to bFLT format
- `--host=arm-elf`
  - Specify the *host* machine that you are building for

Compiling for arm-elf

```
root@slax:~/cis191# head -c 4 /usr/bin/tar && echo
ELF
root@slax:~/cis191# /usr/bin/tar
/usr/bin/tar: You must specify one of the `-Acdtrux' options
Try `/usr/bin/tar --help' or `/usr/bin/tar --usage' for more information.
root@slax:~/cis191# head -c 4 ./tar-1.26/src/tar && echo
bFLT
root@slax:~/cis191# ./tar-1.26/src/tar
-bash: ./tar-1.26/src/tar: cannot execute binary file
root@slax:~/cis191#
```

- Result:
  - Native tar is in `ELF` format, cross-compiled is `bFLT` format
  - Native tar can execute, cross-compiled tar can't (not on the build computer at least)

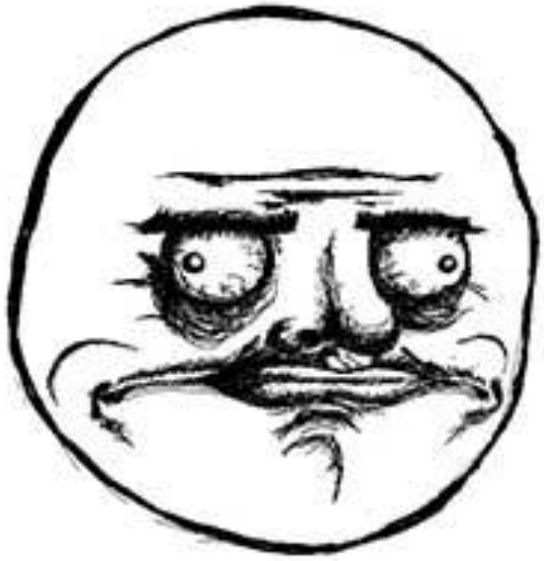
Cross-compiling Terminology

- Note on compiling terms: build, host, target
- **Build**: *platform* that you are building *on*
  - Usually unspecified (since almost always Linux)
- **Host**: *platform* that you are building *for*
  - For cross-compiling, e.g. `arm-elf` architecture
- **Target**: *machine* that you are building *for*
  - For cross-compiling, only specified for special cases with different output formats

That was easy!

- Review: To cross-compile Linux for a *supported* platform, just add a few `config` flags, and run `make`!
- That was easy!

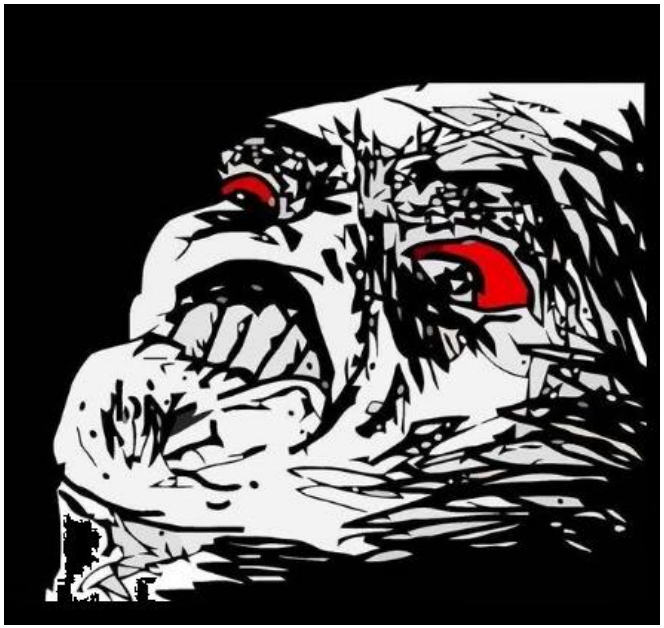
=



But wait, there's more!

- But what happens if you want to run Linux on an *unsupported* platform?
- Too bad, you'll have to port it yourself!

=



Porting Linux = Hard

- Porting Linux in a nutshell:
  1. Gather as much information about the hardware
  2. Reverse-engineer any currently existing software
  3. Modify the cross-compiling tools to generate binaries compatible with the new architecture
  4. Modify the kernel source code to support communicating with the various hardware components
  5. LFS all-over-again! (Except it probably won't work the first time, or even the second)

Porting Linux = Hard

- Porting Linux *minimum* requirements:
  - C programming
  - Linux (CIS 191)
  - Compilers (CIS 341)
  - OS concepts (CIS 380)
  - Computer architecture (CIS 501)
  - Experience with hardware debugging (e.g. JTAG)
  - In-depth knowledge of the assembly language of the target architecture (e.g. x86, ARM, MIPS, etc.)

Why bother?

So you can do this

Case Study: iPodLinux

What is iPodLinux

- iPodLinux = iPod + Linux
  - Custom port of µClinux to the old iPod hardware
  - Goal to turn your iPod into more than just an MP3 player
  - Real reason: Because we can!
  - Wiki: http://ipl.derpapst.eu
    IRC: #ipodlinux@irc.freenode.net
    Code: http://sourceforge.net/projects/ipodlinux/

The Features - Software

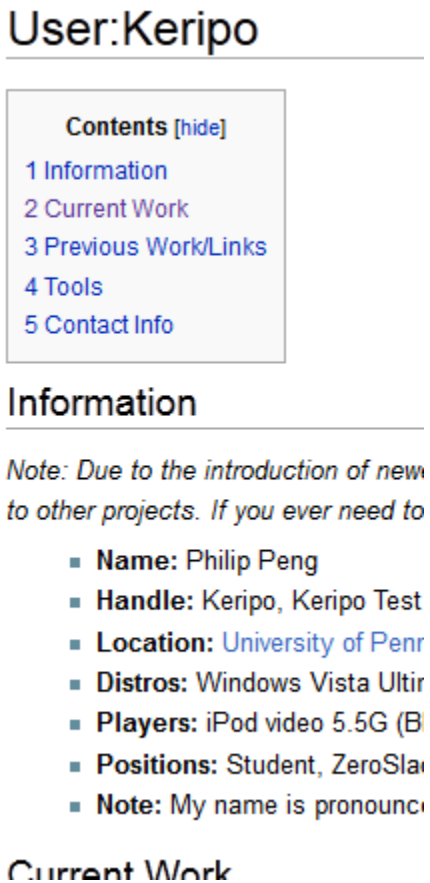
- Customizable user interface
- File-browser and plugin support
- Music player w/ OGG & FLAC support
- Video playback with sound
- Many user-ported Linux applications and emulators

The Features - Hardware

- Custom graphical bootloader
- Playback of audio with piezo (scroll “clicker”)
- Audio-recording via headphone jack
- Backlight brightness control
- Overclocking CPU to 80MHz (vs Apple’s 66MHz)

My contribution

- Joined official dev team in 2008
  - Free iPod gift? Lets hack it!
  - Sansa e200 kernel patches
  - podzilla2 features + bug fixes
  - Experimental kernel builds
  - Compiling tutorials + tools
  - Wiki and forum maintenance



User:Keripo

Contents [hide]

1 Information
2 Current Work
3 Previous Work/Links
4 Tools
5 Contact Info

Information

Note: Due to the introduction of new
to other projects. If you ever need to

- **Name:** Philip Peng
- **Handle:** Keripo, Keripo Test
- **Location:** University of Penr
- **Distros:** Windows Vista Ultir
- **Players:** iPod video 5.5G (B
- **Positions:** Student, ZeroSla
- **Note:** My name is pronounc

Current Work

- See http://ipl.derpapst.eu/wiki/User:Keripo

My contribution

- Project ZeroSlackr
  - Custom, non-destructive iPL installation system
  - Ported numerous third part applications:
    - igpSP – Gameboy Advanced emulator
    - hDoom – original Doom video game
    - hWolf3D – original Wolfenstein3D
    - … and much more
- See http://sourceforge.net/projects/zeroslackr/

History Bit: Reverse Engineering

- Problem:
  No source code/documentation
- Solution:
  Reverse engineer it!
  - Software not encrypted, can be dumped through hardware means
  - Apple left in a Diagnostic Mode
  - iPodLinux project goes live in 2003

More: http://web.archive.org/web/20070224164831/http://www.ipodlinux.org/blog/

History Bit: Piezo Hack

- Problem:
  Can't dump iPod 4G bootloader
- Solution:
  Record it bit-by-bit!
  - Use the piezo ("clicker") to read the bootloader code as sound
  - Put iPod in sound-proof chamber
  - Leave iPod on overnight, decode the audio recording the next day

More : http://web.archive.org/web/20070519081643/http://www.ipodlinux.org/stories/piezo/

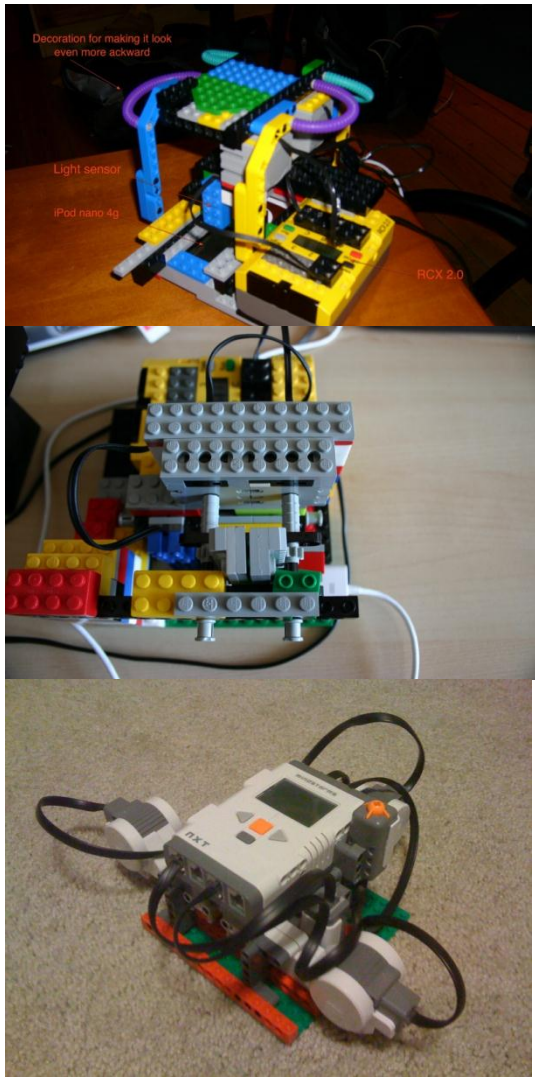
History Bit: We had video first!

- Problem:
  Still pictures on Apple's new iPod
  Photo is boring, 2005
- Solution:
  Lets add video support!
  - Uncompressed, 15fps, A/V issues
  - Apple responds a year later with the iPod 5G, the "iPod video" ; (
  - We did it first! Still counts!

More : http://ipl.derpapst.eu/wiki/Video_Player

History Bit: Nanotron 3000

- Problem:
  iPod nano 2G encrypted, 2006
- Solution:
  Find an exploit!
  - Buffer overflow in Notes functionality (no bound check beyond 268 chars in <a href> links)
  - Use LEGO Mindstorm to brute-force the jump address location



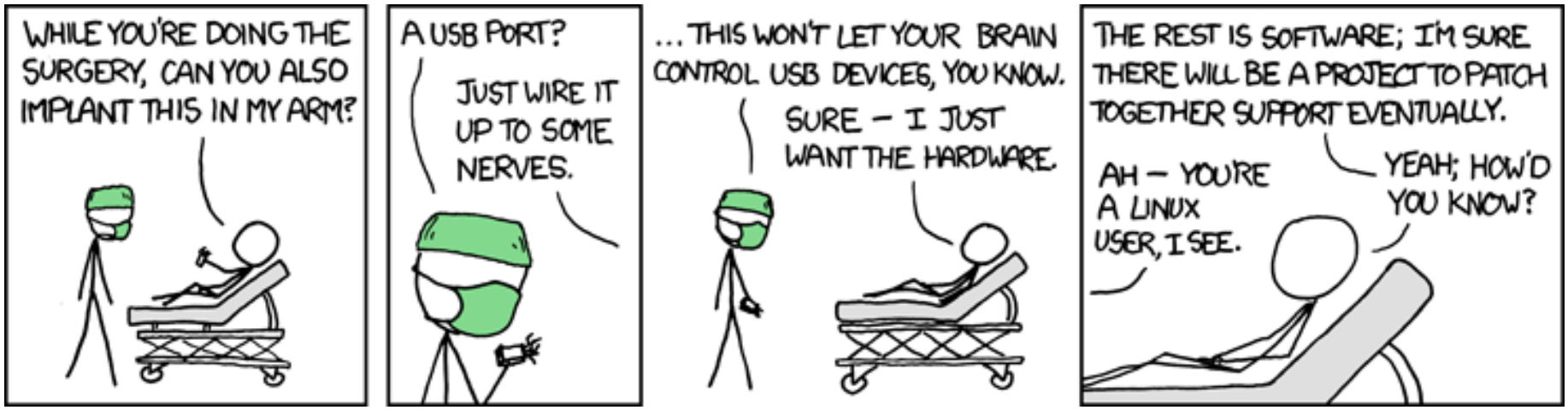
More : http://www.freemyipod.org/wiki/Nanotron_3000

History Bit: iPhone

- Problem:
  Apple releases iPhone
  and iPod Touch in 2007
- Solution:
  None, it was a good run ; (
  - Go work on other cool projects!
  - davidc (David Carne) worked on jailbreakme.com
  - AriX (Ari Weinstein) worked on iJailbreak
  - I work on Android and TA this course (Linux!)

More: http://www.tuaw.com/2007/10/29/instant-jailbreak-for-iphone-and-ipod-touch/
More: http://online.wsj.com/article/SB124692204445002607.html

Questions?


WHILE YOU'RE DOING THE SURGERY, CAN YOU ALSO IMPLANT THIS IN MY ARM?
A USB PORT?
JUST WIRE IT UP TO SOME NERVES.
...THIS WON'T LET YOUR BRAIN CONTROL USB DEVICES, YOU KNOW.
SURE – I JUST WANT THE HARDWARE.
THE REST IS SOFTWARE; I'M SURE THERE WILL BE A PROJECT TO PATCH TOGETHER SUPPORT EVENTUALLY.
AH – YOU'RE A LINUX USER, I SEE.
YEAH; HOW'D YOU KNOW?