Location of Files

- notice where “make install” wanted to put the executable:

Location of Files

- notice where “make install” wanted to put the executable:

  `/usr/local/bin`

  This is another common FHS location

Location of Files

- notice where "make install" wanted to put the executable:

  `/usr/local/bin`

  This is another common FHS location
- also note that most packages produce more than one file; for example:
  - one or more executable files
  - perhaps some libraries
  - a manual or help files
  - perhaps some run-time data files

  "make install" knows where to put these files as well

Demo: Compile file(1)

- home page: http://www.darwinsys.com/file/
- (switch to Linux)

Demo: Compile file(1)

- home page: http://www.darwinsys.com/file/
- (switch to Linux)
- what was different?
  we used –prefix=/path/to/install on the ./configure step

Compiling for PowerPC

now that we know how to compile an GNU application, how do we compile a GNU application for a Platform FPGA?

Compiling for PowerPC

now that we know how to compile an GNU application, how do we compile a GNU application for a Platform FPGA?

- it would be awkward to try and copy it into XPS

Compiling for PowerPC

now that we know how to compile an GNU application, how do we compile a GNU application for a Platform FPGA?

- it would be awkward to try and copy it into XPS
- when we use `gcc` we are creating an Intel x86 executable — we want a PowerPC executable!

Compiling for PowerPC

now that we know how to compile an GNU application, how do we compile a GNU application for a Platform FPGA?

- it would be awkward to try and copy it into XPS
- when we use `gcc` we are creating an Intel x86 executable — we want a PowerPC executable!
- if we had `gcc` on the Platform FPGA (and all of the build tools) we could compile the application there
  - very slow
  - chicken-and-egg problem? how do we get started?

Cross-Compiling

- in other words, we essentially have two systems we need to deal with:
  1. the system that we use to compile the application (the build machine)
  2. the system that will run the application (the host machine)
- ***cross-compilation*** is when the compiler tools on the build system produce an executable for a different (host) system

Build, Host, and Target

- $\boxed{\textit{host}}$ — the machine that executes the application
- $\boxed{\textit{build}}$ — the machine that compiles (links, etc) the application
- for some applications (compilers, debuggers, disassemblers), there is a third term:
  $\boxed{\textit{target}}$ — the machine that the application will generate output for

Example 1: Renesas QSK26A Application

- If you took the Embedded Systems course, then you have used a cross-compiler.
  - the application is developed on a PC (with HEW)
  - resulting executable is run on the M16C microcontroller unit

Example 1: Renesas QSK26A Application

- If you took the Embedded Systems course, then you have used a cross-compiler.
  - the application is developed on a PC (with HEW)
  - resulting executable is run on the M16C microcontroller unit
- in our terms,
  - the PC is the *build* machine
  - the M16C is the *host* machine

Example 2: A Cross-Compiler

- Assume we want to compile gcc so that it runs on a SPARC/Solaris workstation and produces PowerPC 405 executables.
- Moreover, we want to compile gcc on homer machine because it is faster.
- Then...

Example 2: A Cross-Compiler

- Assume we want to compile gcc so that it runs on a SPARC/Solaris workstation and produces PowerPC 405 executables.
- Moreover, we want to compile gcc on homer machine because it is faster.
- Then...
  - the homer is the *build* machine
  - the SPARC/Solaris is the *host* machine
  - the PowerPC 405 is the *target* machine

Naming Machines

So how many different kinds of systems are out there?

- Well, different hardware...

Naming Machines

So how many different kinds of systems are out there?

- Well, different hardware...
  - Sun SPARC

Naming Machines

So how many different kinds of systems are out there?

- Well, different hardware...
  - Sun SPARC
  - HP's PA-RISC

Naming Machines

So how many different kinds of systems are out there?

- Well, different hardware...
  - Sun SPARC
  - HP's PA-RISC
  - Intel's x86, IA-32, and IA-64

Naming Machines

So how many different kinds of systems are out there?

- Well, different hardware...
  - Sun SPARC
  - HP's PA-RISC
  - Intel's x86, IA-32, and IA-64
  - AMD x86_64

Naming Machines

So how many different kinds of systems are out there?

- Well, different hardware...
  - Sun SPARC
  - HP's PA-RISC
  - Intel's x86, IA-32, and IA-64
  - AMD x86_64
  - IBM PowerPC

Naming Machines

So how many different kinds of systems are out there?

- Well, different hardware...
  - Sun SPARC
  - HP's PA-RISC
  - Intel's x86, IA-32, and IA-64
  - AMD x86_64
  - IBM PowerPC
  - ⋮

Naming Machines

So how many different kinds of systems are out there?

- Well, different hardware...
  - Sun SPARC
  - HP's PA-RISC
  - Intel's x86, IA-32, and IA-64
  - AMD x86_64
  - IBM PowerPC
  - ⋮
- And different vendors...

Naming Machines

So how many different kinds of systems are out there?

- Well, different hardware...
  - Sun SPARC
  - HP's PA-RISC
  - Intel's x86, IA-32, and IA-64
  - AMD x86_64
  - IBM PowerPC
  - ⋮
- And different vendors...
- And different operating systems...

Configuration Triple

- Solution is a configuration triplet
  1. CPU (sparc, powerpc, i586, i386)
  2. manufacturer (sun, pc, unknown)
  3. operating system
- Examples: **sparc-sun-solaris2.6**, **i386-pc-winnt4.0**

No Longer a Triple

- Sometimes is obvious to leave out one part
  - **i386-linux** (the manufacturer doesn't matter)
  - **sparc-sunos** (the manufacturer is Sun)
- Other times, more details are necessary
  - an operating system using mostly GNU software with the Linux kernel would be specified: **i586-pc-linux-gnu**
  - an operating system using all GNU software: **i586-pc-hurd-gnu**
- our PowerPC is typically called: **powerpc-405-linux-gnu**

Demo: Cross-Compiler