98-023A : Concurrent and Distributed Programming w/ Inferno and Limbo

Phillip Stanley-Marbell
pstanley@ece.cmu.edu

Lecture Outline

- Inferno Kernel / Emulator Overview
- Source tree and compilation tools
- Example: *Compiling the emulator*

Inferno Source Tree

- Architecture directories
  - /FreeBSD
  - /Hp, etc.
- Libraries
  - /lib9 etc.
- Emulator
  - /emu
- Native Kernels
  - /os

and...

- Inferno root
  - /dev
  - /dis, etc.

Architecture Directories

- For each *system architecture* (e.g., Linux), there may be many different *machine architectures* (e.g., 386, arm, mips, etc.)

```
/Linux/
  386/
    bin/
       asm
       ...
       limbo
       mk
       yacc
  include/
       fpuctl.h
       lib9.h
  lib/
       libbio.a
       ...
       lib9.a
```

Architecture Directories

- Architecture directories contain *host-specific header files* and *libraries* for compiling the emulator, as well as the *host-specific tools* for compiling native kernels

- Example: mk, the compilation / maintenance utility

```
/Linux/
  386/
    bin/
      mk
```

- Example: Libraries which emulator links against

```
/Linux/
  386/
    lib/
      libbio.a
      ...
      lib9.a
```

Emulator Source

- Emulator source resides in `/emu/`

```
/emu/
   MacOSX/
      asm-power.s
      cmd.c
      deveia.c
      devfs.c
      ipif.c
      os.c
```

- Each system architecture directory contains platform specific code for emulator on that host platform
  - Code for creating processes (in `os.c`)
  - Interacting with host's filesystem (`devfs.c`)
  - Accessing host's network protocol stack (`ipif.c`), etc.

Emulator source

- The bulk of the emulator source is architecture independent, and is in `/emu/port/`

```
/emu/
   port/
      audio.h
      ...
      devprog.c
      devssl.c
      win-xll.c
```

  - In general, throughout source tree, architecture independent (or *portable*) code is placed in a directory called `port/`

- Emulator source relies on many routines implemented in the libraries (e.g., `libdraw`, `libinterp`, etc), which are shared with native kernel

Compilation Tools

- Mk
  - The analogue of make on Unix
  - Follows rules defined in a mkfile
  - You'll need to bootstrap a working mk before you can compile any of the tools
  - A working mk is provided for all supported host platforms; Otherwise you can compile manually or from a shell script

Compilation: Configuration Files

- All mkfiles include the configuration file `/mkconfig`

```
<../../mkconfig  # Pull in mkconfig from two steps below in tree
```

- `mkconfig` defines many variables used throughout
  - `ROOT` : This defines the location of the root of your inferno source tree
  - `SYSHOST` : The system architecture of host machine
  - `SYSTARG` : The system architecture that is is being compiled for. When building the emulator, this is identical to `SYSHOST`. When building native kernel, this is set to `"Inferno"`
  - `OBJTYPE` : The machine architecture of the target machine (as defined in `SYSTARG`)
- `mkconfig` includes two additional files which define compiler to use, etc.

Compilation: Configuration Files

- `mkconfig` includes two more configuration files, based on `SYSHOST`, `SYSTARG` and `OBJTYPE`

/mkfiles/mkhost-$SYSHOST

- e.g., /mkfiles/mkhostMacOSX. Defines things like

```
AWK = gawk
CP = cp
SHELLNAME = /bin/sh
```

/mkfiles/mkfile-$SYSTARG-$OBJTYPE

- e.g., /mkfiles/mkfile-MacOSX-power

```
AS = gcc -c
CC = gcc -c
CFLAGS = -arch ppc -Wno-long-double -I$ROOT/MacOSX/power/include
```

etc.

Configuration: Emulator Configuration files

- These defines what files to build into emulator
- Parsed my shell scripts to generate a c source file and a header file
- More on these when we talk about kernel config. files...

Example: Compiling the Emulator

Next

- Emulator structure and data structures
- Kernel structure and data structures
- Getting Inferno to run on a Linksys wrt54g 802.11g wireless router

Fin.