Embedded Applications

What to Expect?

- W's of Embedded Applications
- Various OSS Applications
- Design Care for ES Apps
- Programming Approaches for ES Apps

Application Requirements in ES

- As any other System, features of ES are also used,
  - ultimately by its applications, which can be categorized as follows:
- Start-up Applications
  - Brings up services
  - Sets up network
  - Issues log-in prompt
  - And optionally provides shell interface
- Peripheral Interfacing Applications
  - Display
  - Input System
  - Controls, ...
- ES Requirements-specific Software Applications

What's special about ES Apps?

- May have many more constraints than the non-ES ones
  - Smaller Binary Size
  - Limited Library Support
  - Higher Fault Tolerance
  - Time Tolerant (Running forever)
  - Limited or No Console Services
  - Seamless Hardware Interaction
  - Lesser Processor Power
  - Real Time Requirements
  - ...
- In short, they are
  - Minimized, Optimized, Robust, Continuously Performing Software

Development of ES Apps

- Three ways
  - Port the Standard Applications
    - Typically, available as Open Source, e.g. under GNU License
  - Customized Applications
    - Port & Customize the Open Source Applications
    - Also becomes Open Source
  - Proprietary Applications
    - Write your own from Scratch
- All these are Cross Compiled
  - And then transferred to the Target
  - For desired execution

GNU-style Applications

- Configure using ./configure script
  - With options for Cross Compilation
  - Checks the system for its dependencies
    - Tool headers, ...
  - Generates a Makefile for Cross Compiling the Application
    - Using the Cross Toolchain
- Build the Application using make
- Install the Application on the Target (3 ways)
  - Manually, copy the executables & their dependencies
  - Transfer the “build structure” to target & make install
  - make install to the target's RFS on the host

Start-up ES Apps in OSS

- busybox
  - Small footprint shell environment
  - Download from http://www.busybox.net
- udev
  - Dynamic Device Daemon
  - Download from http://kernel.org/pub/linux/utils/kernel/hotplug/udev.html
- emutils
  - diet libc based very basic shell environment
  - Download from http://www.fefe.de/embutils
- Minit
  - diet libc based non System V compliant init system
  - Download from http://www.fefe.de/minit/

busybox – A special mention

- busybox is so-called a Swiss Knife
- Shell environment being just the starting point
- It provides
  - init system as per System V standard
  - Startup applications & Service daemons
  - mdev: Light-weight udev implementation
  - TinyLogin: Set of logging utilities
  - ...
- Building it is similar to any other OSS

busybox Init

- Sets up signal handlers for init
- Initializes console
  - console arg, or /dev/console
- Parses init's configuration file /etc/inittab
- Runs the system initialization script
  - Value of sysinit. Typically, /etc/init.d/rcS
- Runs all inittab commands in the following order
  - wait → that blocks
  - once → run only once & non-blocking
- Loop over commands that need
  - respawn (on terminate) & askfirst (before respawn)

Other Useful OSS Apps

- TinyX
  - X Server for devices tight on memory
  - Part of X Windows source tree
  - Download from ftp://ftp.xfree86.org/pub/XFree86/4.0/
- thttpd
  - Tiny HTTP Server
  - Download from http://www.acme.com/software/thttpd
- gdb & gdbserver
  - Excellent debuggers for Embedded environment
  - Download from http://www.gnu.org/software/gdb
- And many more related to
  - Graphics
  - Audio
  - ...

Proprietary Applications

- Typically, Start-up & standard Peripheral Interfacing Apps
  - Taken from OSS, as is, or with some customization
- But, what if we need a custom ES specific application, Or
- We don't want to make our application OSS, as
  - It contains some confidential info
- Then, we would like to make it proprietary, Or
- More than that, would have to develop ourselves
- And, that is where we have to be really aware about the various design & programming aspects for the ES Apps
- To be specific

Design Care for ES Apps

- Functionality Design & Programming
- Design for Fault Tolerance
- Hardware aware Design & Programming
- Design & Programming for Performance
- Design for Maintainability
- And,

Right Mix of Languages

- Embedded Application needs to do a wide category of Tasks
  - File Operations
  - Pattern Matching
  - Formula Computation
  - GUI Displays
  - Artificial Intelligence
  - Conditional Controls
  - ...
- Any one language is not the best choice for the variety
- Best Design Approach should consider the Right Mix of Languages
  - Perl, Scripting, C, Java, Expect, ...

  to implement the best suited part
- And then combine them using various interfaces

Portable Languages

- Based on the usage & popularity, many other languages are portable today
- Listing the interesting ones
  - C++
  - Perl
  - Python
  - Java
  - PHP
  - ...

What all have we learnt?

- W's of Embedded Applications
  - Functional & Non-Functional Requirements
  - Development Possibilities
- Various OSS Applications
  - busybox – Swiss Knife
- Design Care for ES Apps
  - Functionality
  - Fault Tolerance, Hardware, Performance, Maintainability
- Programming Approaches for ES Apps
  - Right Mix of Languages

Any Queries?

Backup

Get onto the Board
to
Explore the Start-up Apps

Comparison with Desktop init

- Default configuration file: /etc/inittab
- Different level executions separated by
  - “rc” directories: /etc/rc.d/rc[0-6].d
  - Apart from the system level sysinit
- Levels
  - 0 – halt
  - 1 – single user mode
  - 2 – multi-user without networking
  - 3 – multi-user with networking
  - 5 – X11
  - 6 - reboot