Exploring Suitability of Linux for Embedded Vision Applications

Mini Project

Ankit Mathur	2000101
Mayank Agarwal	2000117

Objectives

- Explore suitability of Linux for embedded systems
- Explore ways to tailor Linux for a specific application
- Implement a miniature customizable running setup
- Implement an embedded vision surveillance application on the setup

Requirements of embedded systems

- Applications of the system are limited and fixed beforehand
- Optimizing the resources used - memory, flash, etc critical to success of the product
- Full functionality of a workstation not needed

Why embed Linux?

- Large support for various architectures, devices, file systems
- Open source
- Stable, scalable
- Familiar, easily customizable
- Large amounts of source code available

Steps in miniaturizing Linux

- Pruning the Kernel
- Selecting a suitable replacement for GlibC
- Providing shell and other command-line tools needed for the application
- Including other support – device drivers, etc. needed by the application

Miniaturizing the Kernel: Options

- Use a small older distribution like 2.0.36
- Use the latest 2.4 series, compiled with minimal features
- Use other distributions tailored for embedded systems like µClinux

Option: An Older Kernel – 2.0.36

- Extremely small (compiled image around 360 kB)
- Major changes in kernel structure and kernel symbols, large parts of code might need to be recoded to port
- Lacking support for newer File Systems, drivers for newer devices

Option: Working with 2.4 kernel

- Wide driver and file system support
- Lots of open source code available, no compatibility problems in compiling and using

- Rather large (~ 1 MB)
- However, the size can be pushed down to about 600 – 700 kB or even less by selecting only the desired modules

Miniaturizing 2.4 Kernel

Configuration of Kernel	Change(in kB)
ext3 (file system) support	40
msdos & vfat support	30
USB device support	30

Options: Specialized Distributions

- Many specialized distributions available e.g. ThinLinux, µClinux
- Extremely small foot-print
- µClinux based on 2.0.36 kernel, provides a patch for it
- No MMU support in µClinux
- Generally, device drivers – a problem

LibC

- Large number of features provided by GlibC can be done away with
- NewLib, µcLibc much better suited for embedded applications
- Our choice - µcLibc

Comparison of binaries

- Size of static binaries using NewLib and µClibC much smaller than those with GlibC
- The standard “ hello world” applications show a massive difference of more than1200 %

```
$ ls –l
-rwxr-xr-x    1 root     root          364k Mar 22 23:54 a.out.glibc
-rwxr-xr-x    1 root     root           30k Mar 22 23:53
a.out.newlib
-rwxrwxr-x    1 minip    minip          18k Mar 23 00:19
a.out.uclibc
```

Busybox

- A single binary providing most of the commonly used command-line tools
- Options to choose the required functions by editing the file “ Config.h”
- Replaces all commonly used command-tools with one single executable, thus achieving massive miniaturization

Busybox (cont’ d)

- Make sym-links from required commands to busybox
- Easily compiled with µcLibC
- Static binary (with µcLibC) size only ~250 kB

A Demo Application

- 2.4 Kernel + Busybox + TinyLogin
- A chat server that allows users on remote machines to chat with each other
- TinyLogin provides Login and User management facilities in a single binary

Conclusion

- Linux well-suited for our embedded application
- The following combination found suitable
  - 2.4 Kernel
  - µclibC
  - Busybox
- Option for shared library/static linking -Depending on the application

Future Targets

- Vision – based surveillance application
- Target architecture – x86 based VIA board
- Targeted memory –
  - 16/32 MB flash
  - 16/32/64 MB RAM
- USB support for camera required
- Network (ethernet) support for communication with server required