Embedded COMPUTING DESIGN

Printed from:
http://www.embedded-computing.com/departments/eclipse/2006/11/

Easing embedded Linux software development for SBCs

By Nathan Gustavson
EMAC, Inc.

and Eric Rossi
EMAC, Inc.

Most programmers today leaving college with Electrical Engineering or Computer Science degrees are fairly well versed in programming on a Linux desktop platform. When these programmers need to take the leap to programming embedded Linux systems, they are often surprised and bewildered that the program they wrote on their desktop will not run their Linux embedded system.

To alleviate this problem, some programmers choose to use versions of desktop systems in embedded applications. This is generally not a good fit as desktop systems typically have fans and hard drives, which are failure points. A bigger obstacle is *fast change*; the need to have the fastest, newest technology obsoletes desktop systems in as little as six months. Designing a long-life embedded product around a platform with no longevity is a recipe for disaster.

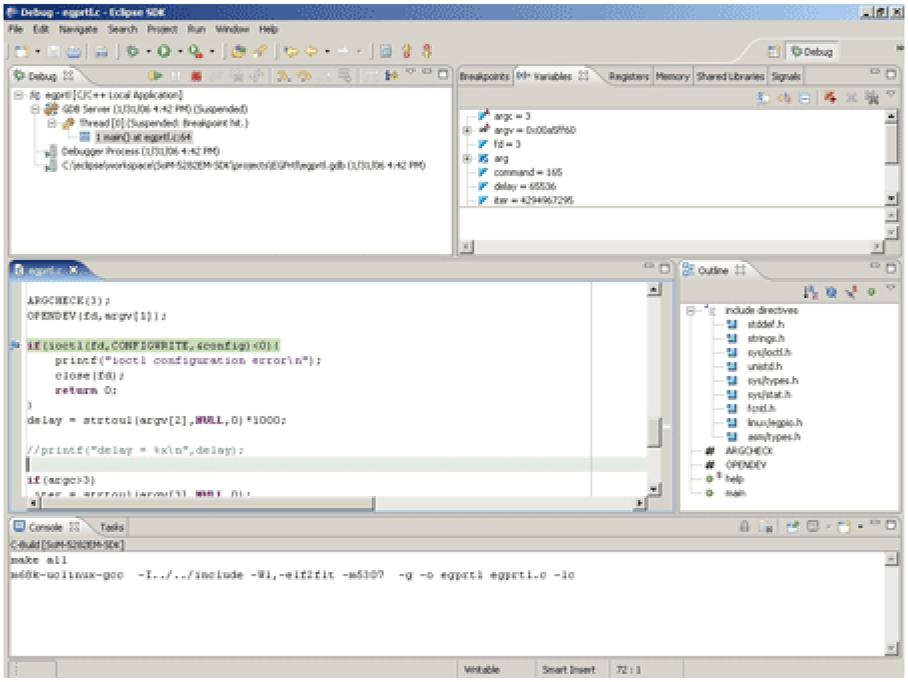
Even if desktop platforms could be guaranteed to be around five years from now, the Linux kernel and associated distributions are moving targets. The kernel and standard libraries are constantly evolving, and distributions are fighting to keep pace. Kernel patches and library updates can render a user's program inoperable. This situation is exacerbated when engineers are required to update development machines due to a security hole or other issue. A way to abstract the embedded development environment from the standard desktop environment is needed.

The key is not only to find a long-life embedded Linux hardware platform, but also a long-life embedded Linux software platform. Accomplished embedded Linux programmers often can create an embedded development platform by gathering the correct libraries and tool versions, and modifying the Makefile appropriately. Even when successful, it's a suboptimum development environment.

Ideally, compilers should not rely on the libraries and kernels of their host, but instead on the tool chain of the more static target. The software platform should be compatible with other platforms, other Operating Systems (OSs), and other languages. The Software Development Kit (SDK) should be based on tool chains with broad community support so that even if one of the organizations driving the SDK stops supporting it, it survives on the shoulders of the rest.

Enter Eclipse

Eclipse is an open source, redistributable, cross-platform, framework for code development. It natively supports Java, and C/C++ with its CDT plug-in. It provides many of the same functions that have been reinvented over and over by the commercial Integrated Development Environments (IDEs) (syntax highlighting, language aware searches, code completion, and so on) but doesn't rely on any one particular set of tools to accomplish it (see Figure 1).


Debug - egpprtl.c - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Debug
Breakpoints Variables Registers Memory Shared Libraries Signals
Outline
Console Tasks
Writable Smart Insert 72 : 1

```
egprtl [C/C++ Local Application]
  GDB Server (1/31/06 4:42 PM) (Suspended)
    Thread [0] (Suspended: Breakpoint hit.)
      1 main() at egprtl.c:64
  Debugger Process (1/31/06 4:42 PM)
  C:/eclipse/workspace/SoM-5282EM-SDK/projects/EGPrtl/egprtl.gdb (1/31/06 4:42 PM)
```

```
argc = 3
argv = 0x00a5ff60
fd = 3
arg
command = 165
delay = 65536
iter = 4294967295
```

```
ARGCHECK(3);
OPENDEV(fd,argv[1]);

if(ioctl(fd,CONFIGWRITE,&config)<0){
    printf("ioctl configuration error\n");
    close(fd);
    return 0;
}
delay = strtoul(argv[2],NULL,0)*1000;

//printf("delay = %x\n",delay);

if(argc>3)
```

```
include directives
  stddef.h
  strings.h
  sys/ioctl.h
  unistd.h
  sys/types.h
  sys/stat.h
  fcntl.h
  linux/egpio.h
  asm/types.h
ARGCHECK
OPENDEV
help
main
```

```
C-Build [SoM-5282EM-SDK]
make all
m68k-uclinux-gcc  -I../../include -Wl,-elf2flt -m5307  -g -o egprtl egprtl.c -lc
```

Figure 1 (click to zoom)

The flexibility of this solution led EMAC and other embedded software companies to build development platforms in an Eclipse environment. This commercial adoption has created de facto standardization for embedded code developers. OEMs can write Eclipse plug-ins and projects that work with GNU tool chains while providing compatibility with numerous commercial platforms running both Windows and Linux.

Embedded Java

EMAC's original Eclipse environment was developed for the SoM-400M module, using a DS400 TINI processor from Dallas/Maxim. This processor's firmware contains a proprietary embedded JRE. Building Java code for the system is completed by compiling classes for a normal JRE environment, replacing the core jars with TINI specific jars and then converting the resulting classes into TINI executables.

To do this developers took advantage of Eclipse's native Ant support, which can be used to create a kind of advanced Java Makefile in XML format. Ant then builds targets by using Eclipse's JDTcompileradaptor class, which makes the internal compiler available to Ant's tasks. The Sourceforge TiniAnt extension then performs the final conversion from Java class to TINI executable. All the complex declarations required to do this are hidden in a top-level build.properties file, vastly reducing the complexity of compilation for the SoM-400M to simple, standard Ant tasks.

Cross-compiled C

The Eclipse tools project provides an open, redistributable C development plug-in called the CDT. It provides, among other things:

- Syntax highlighting
- Code completion
- On-the-fly builds
- Debugger integration (using GDB's MI interface)

It does not provide the actual compiler, linker, and debugger binaries. These are specific to the target being built for and must be provided from an external source. They can then be integrated into Eclipse by writing an Eclipse plug-in, or, more simply, by using "standard" make projects and specifying the compiler in the Makefile.

The Makefile in Listing 1 will compile a simple Hello World example project in the EMAC Eclipse environment.

```
SDKBASE=../../
CROSS=$(SDKBASE)gcc-4.0.0-i486-D/bin/i486-linux-
CC=$(CROSS)gcc


LIBFLAGS =-lc
#VERBOSE=-v
CFLAGS= -g
OFLAGS=-Wl


TARGET=hello
CFILES=hello.c


all: $(TARGET)


$(TARGET): objects
$(CC) $(VERBOSE) *.o $(OFLAGS) $(SLIBS)-o $(TARGET) $(LIBFLAGS)


objects: $(CFILES)
$(CC) $(VERBOSE) $(CFLAGS) -c $(CFILES)


clean:
$(RM) *.o *.gdb $(TARGET)


TARGET_IP=192.168.0.1
LOGIN=someuser
PASSWORD=somepassword


upload:
wput $(TARGET) ftp://$(LOGIN):$(PASSWORD)@$(TARGET_IP)/../../tmp/$(TARGET)
```

Like the embedded Java environment, complex compiler/linker flags common to all projects are hidden in a higher-level file, which is then included by all Makefiles in the SDK.

EMAC provides cross-compilers for all its boards by building freely available GNU cross-compilers against the libraries of the target development OS. On Linux these libraries are native; on Windows the Cygwin library is used. We then “hide” the actual Eclipse executable and replace it with a batch file that temporarily adds the required paths to support the GNU tool chains when Eclipse is started.

Debugging

Debugging takes place using GDB and GDBServer to facilitate a remote debugging environment. To accomplish this, the compiled executable is uploaded to the SBC and a GDBServer session is initiated listening on a specific port or serial terminal. The Eclipse Debugging perspective allows the user to then debug the application by specifying the connection parameters to use as well as the version of GDB built for the target architecture.

The Debugging perspective displays the source code of the program to be debugged and the generated assembly code. It is easy to manage and monitor breakpoints and variables, as well as single-step through the program line by line. The current position in the source code is automatically highlighted. Eclipse provides a graphical debugging environment that greatly simplifies the process of remote target debugging.

“With a little integration, an OEM can create a robust, cross-platform development platform for their boards that will run on the same platform adopted by many commercial software vendors.”

Target communication

When developing for an embedded environment, it will be necessary to establish a connection with the target board, either by a serial terminal program or through TCP using a Telnet, SSH, or other similar connection. When this is done through the command line interface or a graphical front end, the programmer is forced to constantly switch between windows to monitor the connection and the IDE simultaneously.

EMAC eliminates this problem by integrating Eclipse plug-ins that allow for SSH, Telnet, and Terminal connections all within Eclipse. This allows the programmer to monitor the embedded target board as if it were a local file system and issue commands directly without having to leave the Eclipse environment.

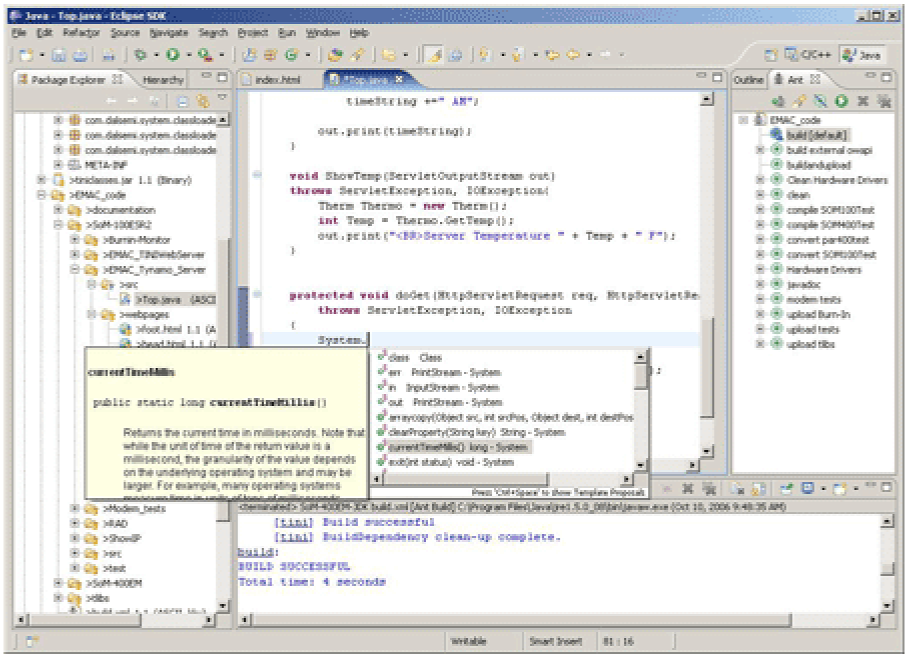
CVS

Eclipse provides a powerful CVS natively. By setting up a CVS login and providing passwords to customers, developers can insure they always have access to the most recent versions of code and bug fixes for products.

Uploading software patches and entirely new components for SDKs onto a CVS server provides up-to-the-minute software support to customers, who can graphically view differences in files and patch their own SDK with some, all, or none of what has been uploaded.

Documentation

Eclipse provides its own native browser, which is beneficial for creating HTML linked documentation right in the workspace. In this way, customers can browse through HTML documentation (similar to that generated by Doxygen and Javadoc). Context-sensitive help is also integrated in Eclipse (see Figure 2), allowing new users to be more productive.



```
        timeString +=" AM";

    out.print(timeString);
}

void ShowTemp(ServletOutputStream out)
throws ServletException, IOException{
    Therm Thermo = new Therm();
    int Temp = Thermo.GetTemp();
    out.print("<BR>Server Temperature " + Temp + " F");
}


protected void doGet(HttpServletRequest req, HttpServletRe
    throws ServletException, IOException
{
    System.
```

```
class  Class
err  PrintStream - System
in  InputStream - System
out  PrintStream - System
arraycopy(Object src, int srcPos, Object dest, int destPos
clearProperty(String key)  String - System
currentTimeMillis()  long - System
exit(int status)  void - System
```

currentTimeMillis

`public static long currentTimeMillis()`

Returns the current time in milliseconds. Note that while the unit of time of the return value is a millisecond, the granularity of the value depends on the underlying operating system and may be larger. For example, many operating systems measure time in units of tens of milliseconds.

```
<terminated> SoM-400EM-JDK build.xml [Ant Build] C:\Program Files\Java\jre1.5.0_08\bin\javaw.exe (Oct 10, 2006 9:48:35 AM)
      [tini] Build successful
      [tini] BuildDependency clean-up complete.
build:
BUILD SUCCESSFUL
Total time: 4 seconds
```

Figure 2 (click to zoom)

This documentation can link right back to the software developer's website as well, allowing user documentation to be added dynamically. Users can switch right from code to documentation with a tab, rather than navigating several windows at once.

Integrated tools, better development

Eclipse provides an environment where GNU tools and commercial tools can play together. With a little integration, an OEM can create a robust, cross-platform development platform for their boards that will run on the same platform adopted by many commercial software vendors.

Within the Eclipse framework, EMAC uses cross-compilation to alleviate the problem of abstracting the embedded development environment from the standard desktop environment. Debugging is accomplished using GDB with Eclipse windows for the debugger's input and output. Communication with the target is also integrated into Eclipse through Telnet, FTP, Terminal, and SSH plug-ins. CVS version control and HTML documentation are both accomplished within the Eclipse environment as well.

All of the aforementioned development capabilities rolled up into a standard IDE make for a powerful, efficient development platform that can be used with a small 8-bit Java-based controller and a 32-bit Linux server programmed in C. This is the power of Eclipse.

Nathan Z. Gustavson is a senior systems engineer at EMAC, Inc. specializing in Linux and real-time control. He holds a BS in Electrical Engineering and is currently pursing a MS at Southern Illinois University.

Eric Rossi is a senior manager at EMAC, Inc. where he is responsible for new product development and custom engineering management. Eric has more than 25 years of experience in embedded system design and holds a BS in Electrical Engineering Technology as well as a BS and MS in Computer Science.

To learn more, contact Nathan or Eric at:

EMAC, Inc.
2390 EMAC Way
Carbondale, IL 62902

618-529-4525
Nathan: ngustavson@emacinc.com
Eric: erossi@emacinc.com

www.emacinc.com