Java(tm) Binary Kernel Support for Linux v1.03

Linux beats them ALL! While all other OS's are TALKING about direct
support of Java Binaries in the OS, Linux is doing it!

You can execute Java applications and Java Applets just like any
other program after you have done the following:

1) You MUST FIRST install the Java Developers Kit for Linux.
   The Java on Linux HOWTO gives the details on getting and
   installing this. This HOWTO can be found at:

   ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/Java-HOWTO

   You should also set up a reasonable CLASSPATH environment
   variable to use Java applications that make use of any
   nonstandard classes (not included in the same directory
   as the application itself).

2) You have to compile BINFMT_MISC either as a module or into
   the kernel (CONFIG_BINFMT_MISC) and set it up properly.
   If you choose to compile it as a module, you will have
   to insert it manually with modprobe/insmod, as kmod
   cannot easily be supported with binfmt_misc.
   Read the file 'binfmt_misc.txt' in this directory to know
   more about the configuration process.

3) Add the following configuration items to binfmt_misc
   (you should really have read binfmt_misc.txt now):
   support for Java applications:
   ```
     ':Java:M::\xca\xfe\xba\xbe::/usr/local/bin/javawrapper:'
   ```
   support for executable Jar files:
   ```
     ':ExecutableJAR:E::jar::/usr/local/bin/jarwrapper:'
   ```
   support for Java Applets:
   ```
     ':Applet:E::html::/usr/bin/appletviewer:'
   ```
   or the following, if you want to be more selective:
   ```
     ':Applet:M::<!--applet::/usr/bin/appletviewer:'
   ```

   Of course you have to fix the path names. The path/file names given in this
   document match the Debian 2.1 system. (i.e. jdk installed in /usr,
   custom wrappers from this document in /usr/local)

   Note, that for the more selective applet support you have to modify
   existing html-files to contain <!--applet--> in the first line
   ('<' has to be the first character!) to let this work!

   For the compiled Java programs you need a wrapper script like the
   following (this is because Java is broken in case of the filename
   handling), again fix the path names, both in the script and in the
   above given configuration string.

   You, too, need the little program after the script. Compile like
   gcc -O2 -o javaclassname javaclassname.c
   and stick it to /usr/local/bin.

```
   Both the javawrapper shellscript and the javaclassname program
   were supplied by Colin J. Watson <cjw44@cam.ac.uk>.

====================== Cut here ===================
#!/bin/bash
# /usr/local/bin/javawrapper - the wrapper for binfmt_misc/java

if [ -z "$1" ]; then
        exec 1>&2
        echo Usage: $0 class-file
        exit 1
fi

CLASS=$1
FQCLASS=`/usr/local/bin/javaclassname $1`
FQCLASSN=`echo $FQCLASS | sed -e 's/^.*\.\([^.]*\)$/\1/'`
FQCLASSP=`echo $FQCLASS | sed -e 's-\.-/-g' -e 's-^[^/]*$--' -e 's-/[^/]*$--'`

# for example:
# CLASS=Test.class
# FQCLASS=foo.bar.Test
# FQCLASSN=Test
# FQCLASSP=foo/bar

unset CLASSBASE

declare -i LINKLEVEL=0

while :; do
        if [ "`basename $CLASS .class`" == "$FQCLASSN" ]; then
                # See if this directory works straight off
                cd -L `dirname $CLASS`
                CLASSDIR=$PWD
                cd $OLDPWD
                if echo $CLASSDIR | grep -q "$FQCLASSP$"; then
                        CLASSBASE=`echo $CLASSDIR | sed -e "s.$FQCLASSP$.."`
                        break;
                fi
                # Try dereferencing the directory name
                cd -P `dirname $CLASS`
                CLASSDIR=$PWD
                cd $OLDPWD
                if echo $CLASSDIR | grep -q "$FQCLASSP$"; then
                        CLASSBASE=`echo $CLASSDIR | sed -e "s.$FQCLASSP$.."`
                        break;
                fi
                # If no other possible filename exists
                if [ ! -L $CLASS ]; then
                        exec 1>&2
                        echo $0:
                        echo "  $CLASS should be in a" \
                             "directory tree called $FQCLASSP"
                        exit 1
                fi
        fi
        if [ ! -L $CLASS ]; then break; fi
        # Go down one more level of symbolic links
        let LINKLEVEL+=1
        if [ $LINKLEVEL -gt 5 ]; then
                exec 1>&2
                echo $0:
                echo "  Too many symbolic links encountered"
                exit 1
        fi
        CLASS=`ls --color=no -l $CLASS | sed -e 's/^.* \([^ ]*\)$/\1/'`
done

if [ -z "$CLASSBASE" ]; then
        if [ -z "$FQCLASSP" ]; then
                GOODNAME=$FQCLASSN.class
        else
                GOODNAME=$FQCLASSP/$FQCLASSN.class
        fi
        exec 1>&2
        echo $0:
        echo "  $FQCLASS should be in a file called $GOODNAME"
        exit 1
fi

if ! echo $CLASSPATH | grep -q "^\(.*:\)*$CLASSBASE\(:.*\)*"; then
        # class is not in CLASSPATH, so prepend dir of class to CLASSPATH
        if [ -z "${CLASSPATH}" ] ; then
                export CLASSPATH=$CLASSBASE
        else
                export CLASSPATH=$CLASSBASE:$CLASSPATH
        fi
fi

shift
/usr/bin/java $FQCLASS "$@"
====================== Cut here ===================


====================== Cut here ===================
/* javaclassname.c
 *
 * Extracts the class name from a Java class file; intended for use in a Java
 * wrapper of the type supported by the binfmt_misc option in the Linux kernel.
 *
 * Copyright (C) 1999 Colin J. Watson <cjw44@cam.ac.uk>.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <sys/types.h>

/* From Sun's Java VM Specification, as tag entries in the constant pool. */

#define CP_UTF8 1
#define CP_INTEGER 3
#define CP_FLOAT 4
#define CP_LONG 5
#define CP_DOUBLE 6
#define CP_CLASS 7
#define CP_STRING 8
#define CP_FIELDREF 9
#define CP_METHODREF 10
#define CP_INTERFACEMETHODREF 11
#define CP_NAMEANDTYPE 12

/* Define some commonly used error messages */

#define seek_error() error("%s: Cannot seek\n", program)
#define corrupt_error() error("%s: Class file corrupt\n", program)
#define eof_error() error("%s: Unexpected end of file\n", program)
#define utf8_error() error("%s: Only ASCII 1-255 supported\n", program);

char *program;

long *pool;

u_int8_t read_8(FILE *classfile);
u_int16_t read_16(FILE *classfile);
void skip_constant(FILE *classfile, u_int16_t *cur);
void error(const char *format, ...);
int main(int argc, char **argv);

/* Reads in an unsigned 8-bit integer. */
u_int8_t read_8(FILE *classfile)
{
        int b = fgetc(classfile);
        if(b == EOF)
                eof_error();
        return (u_int8_t)b;
}

/* Reads in an unsigned 16-bit integer. */
u_int16_t read_16(FILE *classfile)
{
        int b1, b2;
        b1 = fgetc(classfile);
        if(b1 == EOF)
                eof_error();
        b2 = fgetc(classfile);
        if(b2 == EOF)
                eof_error();
        return (u_int16_t)((b1 << 8) | b2);
}

/* Reads in a value from the constant pool. */
void skip_constant(FILE *classfile, u_int16_t *cur)
{
        u_int16_t len;
        int seekerr = 1;
        pool[*cur] = ftell(classfile);
        switch(read_8(classfile))
        {
        case CP_UTF8:
                len = read_16(classfile);
                seekerr = fseek(classfile, len, SEEK_CUR);
                break;
        case CP_CLASS:
        case CP_STRING:
                seekerr = fseek(classfile, 2, SEEK_CUR);
                break;
        case CP_INTEGER:
        case CP_FLOAT:
        case CP_FIELDREF:
        case CP_METHODREF:
        case CP_INTERFACEMETHODREF:
        case CP_NAMEANDTYPE:
                seekerr = fseek(classfile, 4, SEEK_CUR);
                break;
        case CP_LONG:
        case CP_DOUBLE:
                seekerr = fseek(classfile, 8, SEEK_CUR);
                ++(*cur);
                break;
        default:
                corrupt_error();
        }
        if(seekerr)
                seek_error();
}

void error(const char *format, ...)
{
        va_list ap;
        va_start(ap, format);
        vfprintf(stderr, format, ap);
        va_end(ap);
        exit(1);
}

int main(int argc, char **argv)
{
        FILE *classfile;
        u_int16_t cp_count, i, this_class, classinfo_ptr;
        u_int8_t length;
```

```
program = argv[0];

if(!argv[1])
        error("%s: Missing input file\n", program);
classfile = fopen(argv[1], "rb");
if(!classfile)
        error("%s: Error opening %s\n", program, argv[1]);

if(fseek(classfile, 8, SEEK_SET))  /* skip magic and version numbers */
        seek_error();
cp_count = read_16(classfile);
pool = calloc(cp_count, sizeof(long));
if(!pool)
        error("%s: Out of memory for constant pool\n", program);

for(i = 1; i < cp_count; ++i)
        skip_constant(classfile, &i);
if(fseek(classfile, 2, SEEK_CUR))       /* skip access flags */
        seek_error();

this_class = read_16(classfile);
if(this_class < 1 || this_class >= cp_count)
        corrupt_error();
if(!pool[this_class] || pool[this_class] == -1)
        corrupt_error();
if(fseek(classfile, pool[this_class] + 1, SEEK_SET))
        seek_error();

classinfo_ptr = read_16(classfile);
if(classinfo_ptr < 1 || classinfo_ptr >= cp_count)
        corrupt_error();
if(!pool[classinfo_ptr] || pool[classinfo_ptr] == -1)
        corrupt_error();
if(fseek(classfile, pool[classinfo_ptr] + 1, SEEK_SET))
        seek_error();

length = read_16(classfile);
for(i = 0; i < length; ++i)
{
        u_int8_t x = read_8(classfile);
        if((x & 0x80) || !x)
        {
                if((x & 0xE0) == 0xC0)
                {
                        u_int8_t y = read_8(classfile);
                        if((y & 0xC0) == 0x80)
                        {
                                int c = ((x & 0x1f) << 6) + (y & 0x3f);
                                if(c) putchar(c);
                                else utf8_error();
                        }
                        else utf8_error();
                }
                else utf8_error();
        }
```

```
                else if(x == '/') putchar('.');
                else putchar(x);
        }
        putchar('\n');
        free(pool);
        fclose(classfile);
        return 0;
}
====================== Cut here ===================
```

```
====================== Cut here ===================
#!/bin/bash
# /usr/local/java/bin/jarwrapper - the wrapper for binfmt_misc/jar

java -jar $1
====================== Cut here ===================
```

Now simply chmod +x the .class, .jar and/or .html files you want to execute.
To add a Java program to your path best put a symbolic link to the main
.class file into /usr/bin (or another place you like) omitting the .class
extension. The directory containing the original .class file will be
added to your CLASSPATH during execution.

To test your new setup, enter in the following simple Java app, and name
it "HelloWorld.java":

```
        class HelloWorld {
                public static void main(String args[]) {
                        System.out.println("Hello World!");
                }
        }
```

Now compile the application with:

```
        javac HelloWorld.java
```

Set the executable permissions of the binary file, with:

```
        chmod 755 HelloWorld.class
```

And then execute it:

```
        ./HelloWorld.class
```

To execute Java Jar files, simple chmod the *.jar files to include
the execution bit, then just do

```
       ./Application.jar
```

To execute Java Applets, simple chmod the *.html files to include
the execution bit, then just do

```
        ./Applet.html
```

originally by Brian A. Lantz, brian@lantz.com

heavily edited for binfmt_misc by Richard Günther
new scripts by Colin J. Watson <cjw44@cam.ac.uk>
added executable Jar file support by Kurt Huwig <kurt@iku-netz.de>