binfmt_misc.txt

Kernel Support for miscellaneous (your favourite) Binary Formats v1.1

This Kernel feature allows you to invoke almost (for restrictions see below)
every program by simply typing its name in the shell.
This includes for example compiled Java(TM), Python or Emacs programs.

To achieve this you must tell binfmt_misc which interpreter has to be invoked
with which binary. Binfmt_misc recognises the binary-type by matching some bytes
at the beginning of the file with a magic byte sequence (masking out specified
bits) you have supplied. Binfmt_misc can also recognise a filename extension
aka '.com' or '.exe'.

First you must mount binfmt_misc:

```
        mount binfmt_misc -t binfmt_misc /proc/sys/fs/binfmt_misc
```

To actually register a new binary type, you have to set up a string looking like
:name:type:offset:magic:mask:interpreter:flags (where you can choose the ':' upon
your needs) and echo it to /proc/sys/fs/binfmt_misc/register.
Here is what the fields mean:

- 'name' is an identifier string. A new /proc file will be created with this
  name below /proc/sys/fs/binfmt_misc
- 'type' is the type of recognition. Give 'M' for magic and 'E' for extension.
- 'offset' is the offset of the magic/mask in the file, counted in bytes. This
  defaults to 0 if you omit it (i.e. you write ':name:type::magic...')
- 'magic' is the byte sequence binfmt_misc is matching for. The magic string
  may contain hex-encoded characters like \x0a or \xA4. In a shell environment
  you will have to write \\x0a to prevent the shell from eating your \.
  If you chose filename extension matching, this is the extension to be
  recognised (without the '.', the \x0a specials are not allowed). Extension
  matching is case sensitive!
- 'mask' is an (optional, defaults to all 0xff) mask. You can mask out some
  bits from matching by supplying a string like magic and as long as magic.
  The mask is anded with the byte sequence of the file.
- 'interpreter' is the program that should be invoked with the binary as first
  argument (specify the full path)
- 'flags' is an optional field that controls several aspects of the invocation
  of the interpreter. It is a string of capital letters, each controls a certain
  aspect. The following flags are supported -
  - 'P' - preserve-argv[0]. Legacy behavior of binfmt_misc is to overwrite the
    original argv[0] with the full path to the binary. When this flag is
    included, binfmt_misc will add an argument to the argument vector for
    this purpose, thus preserving the original argv[0].
  - 'O' - open-binary. Legacy behavior of binfmt_misc is to pass the full path
    of the binary to the interpreter as an argument. When this flag is
    included, binfmt_misc will open the file for reading and pass its
    descriptor as an argument, instead of the full path, thus allowing
    the interpreter to execute non-readable binaries. This feature should
    be used with care - the interpreter has to be trusted not to emit
    the contents of the non-readable binary.

```
      'C' - credentials. Currently, the behavior of binfmt_misc is to calculate
            the credentials and security token of the new process according to
            the interpreter. When this flag is included, these attributes are
            calculated according to the binary. It also implies the 'O' flag.
            This feature should be used with care as the interpreter
            will run with root permissions when a setuid binary owned by root
            is run with binfmt_misc.


There are some restrictions:
 - the whole register string may not exceed 255 characters
 - the magic must reside in the first 128 bytes of the file, i.e.
   offset+size(magic) has to be less than 128
 - the interpreter string may not exceed 127 characters

To use binfmt_misc you have to mount it first. You can mount it with
"mount -t binfmt_misc none /proc/sys/fs/binfmt_misc" command, or you can add
a line "none  /proc/sys/fs/binfmt_misc binfmt_misc defaults 0 0" to your
/etc/fstab so it auto mounts on boot.

You may want to add the binary formats in one of your /etc/rc scripts during
boot-up. Read the manual of your init program to figure out how to do this
right.

Think about the order of adding entries! Later added entries are matched first!


A few examples (assumed you are in /proc/sys/fs/binfmt_misc):

- enable support for em86 (like binfmt_em86, for Alpha AXP only):
  echo
':i386:M::\x7fELF\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x03:\x
ff\xff\xff\xff\xff\xfe\xfe\xff\xff\xff\xff\xff\xff\xff\xff\xff\xfb\xff\xff:/bin/
em86:' > register
  echo
':i486:M::\x7fELF\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x06:\x
ff\xff\xff\xff\xff\xfe\xfe\xff\xff\xff\xff\xff\xff\xff\xff\xff\xfb\xff\xff:/bin/
em86:' > register

- enable support for packed DOS applications (pre-configured dosemu hdimages):
  echo ':DEXE:M::\x0eDEX::/usr/bin/dosexec:' > register

- enable support for Windows executables using wine:
  echo ':DOSWin:M::MZ::/usr/local/bin/wine:' > register

For java support see Documentation/java.txt


You can enable/disable binfmt_misc or one binary type by echoing 0 (to disable)
or 1 (to enable) to /proc/sys/fs/binfmt_misc/status or /proc/.../the_name.
Catting the file tells you the current status of binfmt_misc/the entry.

You can remove one entry or all entries by echoing -1 to /proc/.../the_name
or /proc/sys/fs/binfmt_misc/status.
```

HINTS:
======

If you want to pass special arguments to your interpreter, you can
write a wrapper script for it. See Documentation/java.txt for an
example.

Your interpreter should NOT look in the PATH for the filename; the kernel
passes it the full filename (or the file descriptor) to use. Using $PATH can
cause unexpected behaviour and can be a security hazard.

There is a web page about binfmt_misc at
http://www.tat.physik.uni-tuebingen.de/~rguenth/linux/binfmt_misc.html

Richard Günther <rguenth@tat.physik.uni-tuebingen.de>