# binfmt_misc

O termo *binfmt_misc* descreve uma capacidade do kernel Linux que permite que formatos de arquivos executáveis arbitrários sejam reconhecidos e passados para certos aplicativos de espaço do usuário, como emuladores e máquinas virtuais.

## Explicação

Os formatos executáveis são registrados por meio de uma interface de sistema de arquivos de propósito especial (semelhante a /proc). O seu kernel deve ser configurado com "`CONFIG_BINFMT_MISC`" embutido ou como um módulo.
O arquivo "`registrar`" contém linhas em um formato específico:

```
:name:type:offset:magic:mask:interpreter:
```

- **name** é o nome do formato binário.
- **type** é **E** ou **M**
  - Se for **E**, o formato do arquivo executável é identificado por sua extensão de nome de arquivo: **magic** é a extensão do arquivo a ser associada ao formato binário; **offset** e **mask** são ignorados.
  - Se for **M**, o formato é identificado por **magic** número em absoluto de **offset** no arquivo e **mask** é uma máscara de bits que indica quais bits no número são significativos.
- **interpreter** é um programa que deve ser executado com o arquivo correspondente como argumento.

## Preparação

Antes de tentar registrar um emulador de programa como wine ou qemu com o kernel como um interpretador, você deve garantir que o *binfmt_misc* o sistema de arquivos está montado e pronto. Adicione a seguinte linha a "`/etc/fstab`" para montar o sistema de arquivos especial:

```
none /proc/sys/fs/binfmt_misc binfmt_misc defaults 0 0
```

Você pode ter que descomentar a linha abaixo em "`/etc/rc.d/rc.modules`":

```
# /sbin/modprobe binfmt_misc
```

Em seguida, use linhas como a abaixo para registrar um novo formato de arquivo binário. Os exemplos aqui permitem que você inicie binários do MS Windows mais facilmente, mas a parte interessante é o formato de arquivo binário ARM. Você deve ser capaz de usar esta configuração para criar uma instalação chroot de ARMedslack em um computador host x86 ou x86_64, e então executar algo como:

```
chroot /path/to/arm_filesystem_root /bin/bash
```

para entrar em um sistema de arquivos ARM emulado sem ser incomodado pelas linhas de comando do qemu.

Normalmente, as seguintes linhas devem ser colocadas no script de inicialização de inicialização local "`/etc/rc.d/rc.local`" porque os formatos binários e seus interpretadores de espaço de usuário devem ser registrados novamente a cada inicialização:

```
# Support for Windows binaries through Wine:
{ echo ':DOSWin:M::MZ::/usr/bin/wine:' > /proc/sys/fs/binfmt_misc/register;
} 2>/dev/null

# Support for ARM binaries through Qemu:
{ echo
':arm:M::\x7fELF\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x28
\x00:\xff\xff\xff\xff\xff\xff\xff\x00\xff\xff\xff\xff\xff\xff\xff\xff\xfe\xf
f\xff\xff:/usr/bin/qemu-arm-static:' > /proc/sys/fs/binfmt_misc/register; }
2>/dev/null
{ echo
':armeb:M::\x7fELF\x01\x02\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x
00\x28:\xff\xff\xff\xff\xff\xff\xff\x00\xff\xff\xff\xff\xff\xff\xff\xff\xff\
xfe\xff\xff:/usr/bin/qemu-armeb-static:' >
/proc/sys/fs/binfmt_misc/register; } 2>/dev/null
```

O caso do qemu como interpretador de binários ARM é um pouco especial. Requer que você use uma *versão compilada estaticamente* da emulação do espaço de usuário ARM do qemu. O executável *qemu-arm* não deve depender de nenhum vínculo dinâmico de bibliotecas host - quando o kernel o inicia, ele se encontrará em um ambiente de nada além do código binário ARM. Ao registrar um novo formato binário com o kernel, ele criará um arquivo em "`/proc/sys/fs/binfmt_misc/`" com detalhes legíveis por humanos para o formato que você acabou de registrar.

# Execução

Uma vez que um formato de arquivo binário foi registrado como este, você não precisa mais iniciar um programa com este formato iniciando primeiro seu emulador. Agora pode ser iniciado diretamente. Por exemplo,

```
wine ~/.wine/dosdevices/c\:/windows/notepad.exe
```

não é mais necessário; é o suficiente para executar

```
~/.wine/dosdevices/c\:/windows/notepad.exe
```

Se você adicionar os diretórios do Wine ao seu $PATH, você poderá até mesmo digitar "`notepad.exe`".

# Apêndice

Aqui está um script **qemu.SlackBuild** que é capaz de construir o binário estático que você precisa para emulação ARM através de *binfmt_misc*.

```
#!/bin/sh
# $Id: qemu-static.SlackBuild,v 1.1 2012/02/20 16:28:29 root Exp root $
# Copyright 2007, 2008, 2009, 2010, 2012  Eric Hameleers, Eindhoven,
Netherlands
# All rights reserved.
#
#   Permission to use, copy, modify, and distribute this software for
#   any purpose with or without fee is hereby granted, provided that
#   the above copyright notice and this permission notice appear in all
#   copies.
#
#   THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED
#   WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
#   MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
#   IN NO EVENT SHALL THE AUTHORS AND COPYRIGHT HOLDERS AND THEIR
#   CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
#   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
#   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
#   USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
#   ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
#   OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
#   OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
#   SUCH DAMAGE.
# ------------------------------------------------------------------------
---
#
# Slackware SlackBuild script
# ===========================
# By:         Eric Hameleers <alien@slackware.com>
# For:        qemu
# Descr:      a generic machine emulator and virtualizer
# URL:        http://qemu.org/
# Needs:
# Changelog:
# 1.0.1-1:    18/feb/2012 by Eric Hameleers <alien@slackware.com>
#             * After another long hiatus, let's pick up this build...
#               once more. This time to complement my qemu-kvm package which
#               I limited to KVM supported architecture exclusively.
#
# Run 'sh qemu.SlackBuild' to build a Slackware package.
# The package (.tgz) plus descriptive .txt file are created in /tmp .
# Install using 'installpkg'.
#
# ------------------------------------------------------------------------
---
```

```
# Set initial variables:

PRGNAM=qemu
PRGNAM2=qemu-static
VERSION=${VERSION:-"1.0.1"}
GLIB=${GLIB:-"2.28.8"}
BUILD=${BUILD:-1}
NUMJOBS=${NUMJOBS:" -j4 "}
TAG=${TAG:-alien}

# If you also want static binaries for user emulation, set BUILD_STATIC=YES
# This will create a second package "qemu-static" with user emulation
binaries.
# NOTE:
# This fails with the glibc 1.13 of Slackware 13.37 !
# The static compilation pulls in the glib-2.0 libraries; since Slackware
does
# not include static versions of these, we will have to build them now:
BUILD_STATIC=${BUILD_STATIC:-YES}

# The documentation:
DOCS="CODING_STYLE COPYING* Changelog LICENSE MAINTAINERS README* TODO
VERSION"

# Where do we look for sources?
SRCDIR=$(cd $(dirname $0); pwd)

# Place to build (TMP) package (PKG) and output (OUTPUT) the program:
TMP=${TMP:-/tmp/build}
PKG=$TMP/package-$PRGNAM
PKG2=$TMP/package-${PRGNAM2}
PKGGLIB=$TMP/package-glib_static
OUTPUT=${OUTPUT:-/tmp}

SOURCE[0]="$SRCDIR/${PRGNAM}-${VERSION}.tar.gz"
SRCURL[0]="http://wiki.qemu.org/download/${PRGNAM}-${VERSION}.tar.gz"
USE[0]="YES"

SOURCE[1]="$SRCDIR/glib-${GLIB}.tar.xz"
SRCURL[1]="http://ftp.gnome.org/pub/GNOME/sources/glib/$(echo $GLIB |cut -d.
-f1,2)/glib-${GLIB}.tar.xz"
USE[1]="$BUILD_STATIC"

##
## --- with a little luck, you won't have to edit below this point --- ##
##

# Automatically determine the architecture we're building on:
MARCH=$( uname -m )
if [ -z "$ARCH" ]; then
```

```
  case "$MARCH" in
    i?86)    export ARCH=i486 ;;
    armv7hl) export ARCH=$MARCH ;;
    arm*)    export ARCH=arm ;;
    # Unless $ARCH is already set, use uname -m for all other archs:
    *)       export ARCH=$MARCH ;;
  esac
fi

case "$ARCH" in
  i486)      SLKCFLAGS="-O2 -march=i486 -mtune=i686"
             SLKLDFLAGS=""; LIBDIRSUFFIX=""
             ;;
  x86_64)    SLKCFLAGS="-O2 -fPIC"
             SLKLDFLAGS="-L/usr/lib64"; LIBDIRSUFFIX="64"
             ;;
  armv7hl)   SLKCFLAGS="-O2 -march=armv7-a -mfpu=vfpv3-d16"
             SLKLDFLAGS=""; LIBDIRSUFFIX=""
             ;;
  *)         SLKCFLAGS="-O2"
             SLKLDFLAGS=""; LIBDIRSUFFIX=""
             ;;
esac

# Exit the script on errors:
set -e
trap 'echo "$0 FAILED at line $LINENO!" | tee $OUTPUT/error-${PRGNAM}.log'
ERR
# Catch unitialized variables:
set -u
P1=${1:-1}

# Save old umask and set to 0022:
_UMASK_=$(umask)
umask 0022

# Create working directories:
mkdir -p $OUTPUT          # place for the package to be saved
mkdir -p $TMP/tmp-$PRGNAM # location to build the source
mkdir -p $PKG             # place for the package to be built
rm -rf $PKG/*             # always erase old package's contents
rm -rf $TMP/tmp-$PRGNAM/* # remove the remnants of previous build
rm -rf $OUTPUT/{configure,make,install,error,makepkg}-$PRGNAM.log
                          # remove old log files

# Source file availability:
for (( i = 0; i < ${#SOURCE[*]}; i++ )) ; do
  [ "${USE[$i]}" != "YES" ] && continue
  if ! [ -f ${SOURCE[$i]} ]; then
    echo "Source '$(basename ${SOURCE[$i]})' not available yet..."
    # Check if the $SRCDIR is writable at all - if not, download to $OUTPUT
```

```
    [ -w "$SRCDIR" ] || SOURCE[$i]="$OUTPUT/$(basename ${SOURCE[$i]})"
    if [ -f ${SOURCE[$i]} ]; then echo "Ah, found it!"; continue; fi
    if ! [ "x${SRCURL[$i]}" == "x" ]; then
      echo "Will download file to $(dirname $SOURCE[$i])"
      wget -nv -T 20 -O "${SOURCE[$i]}" "${SRCURL[$i]}" || true
      if [ $? -ne 0 -o ! -s "${SOURCE[$i]}" ]; then
        echo "Fail to download '$(basename ${SOURCE[$i]})'. Aborting the
build."
        mv -f "${SOURCE[$i]}" "${SOURCE[$i]}".FAIL
        exit 1
      fi
    else
      echo "File '$(basename ${SOURCE[$i]})' not available. Aborting the
build."
      exit 1
    fi
  fi
done

if [ "$P1" == "--download" ]; then
  echo "Download complete."
  exit 0
fi

# --- PACKAGE BUILDING ---

echo "++"
echo "|| $PRGNAM-$VERSION"
echo "++"

cd $TMP/tmp-$PRGNAM
echo "Extracting the source archive(s) for $PRGNAM..."
tar -xvf ${SOURCE[0]}
cd ${PRGNAM}-${VERSION}
chown -R root:root .
chmod -R u+w,go+r-w,a+X-s .

echo Building ...

# Compensate for users who forgot to run "su -":
if ! which texi2html 1>/dev/null 2>/dev/null ; then
  export PATH=$PATH:/usr/share/texmf/bin
fi

# The SlackBuild needs VDE, so abort if it was not installed:
if ! pkg-config --exists vdeplug ; then
  echo -e "**\n** VDE is not installed! Please install it first!\n**"
  exit 1
fi

function qemuconfigure () {
```

```
  # If you do not care for all the exotic target platforms, then you can add
  # a subset of targets to the 'configure' block below. For instance,
  # to only build x86_64 and arm targets, change this line:
  # --target-list="" \
  # to:
  # --target-list=x86_64-softmmu,x86_64-linux-user,arm-softmmu,arm-linux-
user \

  ./configure \
    --prefix=/usr \
    --sysconfdir=/etc \
    --mandir=/usr/man \
    --datadir=/usr/share/$PRGNAM \
    --docdir=/usr/doc/$PRGNAM-$VERSION \
    --enable-linux-user \
    $*
}

# First configure the usual non-static build including ALSA and SDL support:
echo -e "**\n** making dynamic...\n**"
export LDFLAGS="$SLKLDFLAGS"
export CFLAGS="$SLKCFLAGS"
export CXXFLAGS="$SLKCFLAGS"
qemuconfigure \
  --enable-system \
  --target-list="" \
  --audio-drv-list=alsa,oss,sdl,esd \
  --audio-card-list=ac97,es1370,sb16,cs4231a,adlib,gus \
  --enable-mixemu \
  --enable-vde \
  2>&1 | tee $OUTPUT/configure-${PRGNAM}.log

# Compile and install into $PKG:
make $NUMJOBS OS_CFLAGS="$SLKCFLAGS" \
  2>&1 | tee $OUTPUT/make-${PRGNAM}.log
make OS_CFLAGS="$SLKCFLAGS" DESTDIR=$PKG install \
  2>&1 |tee $OUTPUT/install-${PRGNAM}.log

if [ "$BUILD_STATIC" = "YES" ]; then

  # Clean up for the next step:
  echo -e "**\n** making clean...\n**"
  make clean

  # Next, configure a static build - here we target non-PC architectures
only.
  # The static binaries can be used in a chroot on the host system:
  echo -e "**\n** making static...\n**"

  ( # Build a static glib-2.0:
```

```
  # Remove the remnants of previous build:
  rm -rf $PKG2 $PKGGLIB
  mkdir -p $PKG2 $PKGGLIB

  cd $TMP/tmp-$PRGNAM
  echo "Extracting the source archive(s) for glib..."
  tar -xvf ${SOURCE[1]}
  cd glib-$GLIB
  chown -R root:root .
  chmod -R u+w,go+r-w,a+X-s .
  CFLAGS="$SLKCFLAGS" \
  ./configure \
    --prefix=/usr \
    --libdir=/usr/lib${LIBDIRSUFFIX} \
    --sysconfdir=/etc \
    --mandir=/usr/man \
    --disable-dynamic \
    --enable-static \
    --build=$ARCH-slackware-linux \
    2>&1 | tee $OUTPUT/configure_static-${PRGNAM}_glib.log

  # Compile glib and install the static library in a temporary location
  # where qemu can pick it up:
  make $NUMJOBS \
    2>&1 | tee $OUTPUT/make_static-${PRGNAM}_glib.log
  make install DESTDIR=$PKGGLIB \
    2>&1 | tee $OUTPUT/install_static-${PRGNAM}_glib.log
)

export CFLAGS="-I$PKGGLIB/usr/include $SLKCFLAGS"
export CXXFLAGS="-I$PKGGLIB/usr/include $SLKCFLAGS"
export LDFLAGS="-L$PKGGLIB/usr/lib${LIBDIRSUFFIX} $SLKLDFLAGS"
export PKG_CONFIG_PATH="$PKGGLIB/usr/lib${LIBDIRSUFFIX}/pkgconfig"

qemuconfigure \
  --disable-system \
  --disable-curses \
  --disable-sdl \
  --disable-bluez \
  --disable-docs \
  --disable-guest-agent \
  --disable-kvm \
  --disable-vde \
  --disable-smartcard \
  --disable-smartcard-nss \
  --disable-vnc \
  --static \
  2>&1 | tee $OUTPUT/configure_static-${PRGNAM}.log

# Compile and install into $PKG2:
```

```
  make $NUMJOBS OS_CFLAGS="$SLKCFLAGS" \
    2>&1 | tee $OUTPUT/make_static-${PRGNAM}.log
  make OS_CFLAGS="$SLKCFLAGS" DESTDIR=$PKG2 install \
    2>&1 |tee $OUTPUT/install_static-${PRGNAM}.log

  # Rename the binaries so that they do not clash with the dynamic binaries:
  for FILE in $PKG2/usr/bin/* ; do
    mv $FILE ${FILE}-static
  done

  # The user mode emulators do no need these bios/keymap files:
  rm -rf $PKG2/usr/share/$PRGNAM
  rmdir $PKG2/usr/share 2>/dev/null || true

  # Add documentation:
  mkdir -p $PKG2/usr/doc/$PRGNAM2-$VERSION
  cp -a $DOCS $PKG2/usr/doc/$PRGNAM2-$VERSION || true
  cat $SRCDIR/$(basename $0) \
    > $PKG2/usr/doc/$PRGNAM2-$VERSION/$PRGNAM.SlackBuild
  chown -R root:root $PKG2/usr/doc/$PRGNAM2-$VERSION
  find $PKG2/usr/doc -type f -exec chmod 644 {} \;

  # Compress the man page(s):
  if [ -d $PKG2/usr/man ]; then
    find $PKG2/usr/man -type f -name "*.?" -exec gzip -9f {} \;
    for i in $(find $PKG2/usr/man -type l -name "*.?") ; do ln -s $(
readlink $i ).gz $i.gz ; rm $i ; done
  fi

  # Add a package description:
  mkdir -p $PKG2/install
  cat $SRCDIR/${PRGNAM2}.slack-desc > $PKG2/install/slack-desc

  # Build the package:
  cd $PKG2
  makepkg --linkadd y --chown n $OUTPUT/${PRGNAM2}-${VERSION}-${ARCH}-
${BUILD}${TAG}.tgz 2>&1 | tee $OUTPUT/makepkg-${PRGNAM2}.log
  cd $OUTPUT
  md5sum ${PRGNAM2}-${VERSION}-${ARCH}-${BUILD}${TAG}.tgz > ${PRGNAM2}-
${VERSION}-${ARCH}-${BUILD}${TAG}.tgz.md5
  cd -
  cat $PKG2/install/slack-desc | grep "^${PRGNAM2}" > $OUTPUT/${PRGNAM2}-
${VERSION}-${ARCH}-${BUILD}${TAG}.txt

fi # End [ "$BUILD_STATIC" = "YES" ]

# Continuing with the default dynamic build:

# Add documentation:
mkdir -p $PKG/usr/doc/$PRGNAM-$VERSION
cp -a $DOCS $PKG/usr/doc/$PRGNAM-$VERSION || true
```

```
cat $SRCDIR/$(basename $0) > $PKG/usr/doc/$PRGNAM-
$VERSION/$PRGNAM.SlackBuild
chown -R root:root $PKG/usr/doc/$PRGNAM-$VERSION
find $PKG/usr/doc -type f -exec chmod 644 {} \;

# Compress the man page(s):
if [ -d $PKG/usr/man ]; then
  find $PKG/usr/man -type f -name "*.?" -exec gzip -9f {} \;
  for i in $(find $PKG/usr/man -type l -name "*.?") ; do ln -s $( readlink
$i ).gz $i.gz ; rm $i ; done
fi

# Strip binaries:
find $PKG | xargs file | grep -e "executable" -e "shared object" \
  | grep ELF | cut -f 1 -d : | xargs strip --strip-unneeded 2> /dev/null ||
true

# Clean up:
find $PKG/usr/share -type f -size 0 -exec rm -f {} \; 2>/dev/null || true

# Add a package description:
mkdir -p $PKG/install
cat $SRCDIR/${PRGNAM}.slack-desc > $PKG/install/slack-desc
cat $SRCDIR/${PRGNAM}.slack-required > $PKG/install/slack-required

# Build the package:
cd $PKG
makepkg --linkadd y --chown n $OUTPUT/${PRGNAM}-${VERSION}-${ARCH}-
${BUILD}${TAG}.tgz 2>&1 | tee $OUTPUT/makepkg-${PRGNAM}.log
cd $OUTPUT
md5sum ${PRGNAM}-${VERSION}-${ARCH}-${BUILD}${TAG}.tgz > ${PRGNAM}-
${VERSION}-${ARCH}-${BUILD}${TAG}.tgz.md5
cd -
cat $PKG/install/slack-desc | grep "^${PRGNAM}" > $OUTPUT/${PRGNAM}-
${VERSION}-${ARCH}-${BUILD}${TAG}.txt
cat $PKG/install/slack-required > $OUTPUT/${PRGNAM}-${VERSION}-${ARCH}-
${BUILD}${TAG}.dep

# Restore the original umask:
umask ${_UMASK_}
```

You'll also need the **qemu-static.slack-desc** file:

```
# HOW TO EDIT THIS FILE:
# The "handy ruler" below makes it easier to edit a package description.
Line
# up the first '|' above the ':' following the base package name, and the
'|'
# on the right side marks the last column you can put a character in.  You
must
# make exactly 11 lines for the formatting to be correct.  It's also
```

```
# customary to leave one space after the ':'.

           |-----handy-ruler------------------------------------------------
------|
qemu-static: qemu-static (a processor emulator)
qemu-static:
qemu-static: QEMU is a FAST! processor emulator using dynamic translation to
qemu-static: achieve good emulation speed. QEMU has two operating modes:
qemu-static: Full system emulation and User mode emulation.
qemu-static:
qemu-static: This package contains statically compiled binaries of only the
qemu-static: User mode emulators, for use inside chroot environments.
qemu-static:
qemu-static:
qemu-static: Home page at http://qemu.org/
```

# Fontes

- Original escrito por Eric Hameleers
- Traduzido PT-BR por: MacgyverPT (Miguel Rosa) em 2020/11/20 17:58 (UTC)
- Referência Externa: Wikipedia

howtos, software, emulator, author alienbob
translated pt, macgyverpt