```
#!/bin/sh
# Copyright Abandoned 1996 TCX DataKonsult AB & Monty Program KB &
Detron HB
# This file is public domain and comes with NO WARRANTY of any kind
# This file shows the changes needed to fix the MySQL startup and
#   PrefPane issues that exist with MacOS X Snow Leopard
# Added text is in red and bold. No text has been deleted.
# This file lives at  /usr/local/mysql/support-files/mysql.server

# MySQL daemon start/stop script.

# Usually this is put in /etc/init.d (at least on machines SYSV R4 based
# systems) and linked to /etc/rc3.d/S99mysql and /etc/rc0.d/K01mysql.
# When this is done the mysql server will be started when the machine is
# started and shut down when the systems goes down.

# Comments to support chkconfig on RedHat Linux
# chkconfig: 2345 64 36
# description: A very fast and reliable SQL database engine.

# Comments to support LSB init script conventions
### BEGIN INIT INFO
# Provides: mysql
# Required-Start: $local_fs $network $remote_fs
# Should-Start: ypbind nscd ldap ntpd xntpd
# Required-Stop: $local_fs $network $remote_fs
# Default-Start:  2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: start and stop MySQL
# Description: MySQL is a very fast and reliable SQL database engine.
### END INIT INFO

# If you install MySQL on some other places than ., then you
# have to do one of the following things for this script to work:
#
# - Run this script from within the MySQL installation directory
# - Create a /etc/my.cnf file with the following information:
#   [mysqld]
#    basedir=<path-to-mysql-installation-directory>
# - Add the above to any other configuration file (for example ~/.my.ini)
#    and copy my_print_defaults to /usr/bin
# - Add the path to the mysql-installation-directory to the basedir variable
#    below.
```

```
#
# If you want to affect other MySQL variables, you should make your
changes
# in the /etc/my.cnf, ~/.my.cnf or other MySQL configuration files.

# If you change base dir, you must also change datadir. These may get
# overwritten by settings in the MySQL configuration files.

basedir=
datadir=

# Default value, in seconds, afterwhich the script should timeout waiting
# for server start.
# Value here is overriden by value in my.cnf.
# 0 means don't wait at all
# Negative numbers mean to wait indefinitely
service_startup_timeout=900

# Lock directory for RedHat / SuSE.
lockdir='/var/lock/subsys'
lock_file_path="$lockdir/mysql"

# The following variables are only set for letting mysql.server find things.

# Set some defaults
mysqld_pid_file_path=
if test -z "$basedir"
then
# bug fix
#  basedir=.
  basedir=/usr/local/mysql
  bindir=./bin
   if test -z "$datadir"
   then
     datadir=/usr/local/mysql/data
  fi
  sbindir=./bin
  libexecdir=./bin
else
   bindir="$basedir/bin"
   if test -z "$datadir"
   then
     datadir="$basedir/data"
```

```
  fi
   sbindir="$basedir/sbin"
   libexecdir="$basedir/libexec"
fi

# datadir_set is used to determine if datadir was set (and so should be
# *not* set inside of the --basedir= handler.)
datadir_set=

#
# Use LSB init script functions for printing messages, if possible
#
lsb_functions="/lib/lsb/init-functions"
if test -f $lsb_functions ; then
  . $lsb_functions
else
  log_success_msg()
  {
    echo " SUCCESS! $@"
  }
  log_failure_msg()
  {
    echo " ERROR! $@"
  }
fi

PATH="/sbin:/usr/sbin:/bin:/usr/bin:$basedir/bin"
export PATH

mode=$1    # start or stop

[ $# -ge 1 ] && shift


other_args="$*"   # uncommon, but needed when called from an RPM
upgrade action
          # Expected: "--skip-networking --skip-grant-tables"
          # They are not checked here, intentionally, as it is the resposibility
          # of the "spec" file author to give correct arguments only.

case `echo "testingc"`,`echo -n testing` in
   *c*,-n*) echo_n=   echo_c=     ;;
   *c*,*)   echo_n=-n echo_c=     ;;
```

```
   *)      echo_n=   echo_c='c' ;;
esac

parse_server_arguments() {
  for arg do
    case "$arg" in
     --basedir=*)  basedir=`echo "$arg" | sed -e 's/^[^=]*=//'`
                bindir="$basedir/bin"
                 if test -z "$datadir_set"; then
                   datadir="$basedir/data"
                 fi
                 sbindir="$basedir/sbin"
                 libexecdir="$basedir/libexec"
      ;;
     --datadir=*)  datadir=`echo "$arg" | sed -e 's/^[^=]*=//'`
                 datadir_set=1
       ;;
     --pid-file=*) mysqld_pid_file_path=`echo "$arg" | sed -e
's/^[^=]*=//'` ;;
      --service-startup-timeout=*) service_startup_timeout=`echo "$arg" |
sed -e 's/^[^=]*=//'` ;;
    esac
  done
}

wait_for_pid () {
  verb="$1"          # created | removed
  pid="$2"           # process ID of the program operating on the pid-file
  pid_file_path="$3" # path to the PID file.

  i=0
  avoid_race_condition="by checking again"

  while test $i -ne $service_startup_timeout ; do

    case "$verb" in
     'created')
       # wait for a PID-file to pop into existence.
       test -s "$pid_file_path" && i='' && break
      ;;
     'removed')
       # wait for this PID-file to disappear
       test ! -s "$pid_file_path" && i='' && break
```

```
      ;;
    *)
      echo "wait_for_pid () usage: wait_for_pid created|removed pid
pid_file_path"
      exit 1
      ;;
  esac

  # if server isn't running, then pid-file will never be updated
  if test -n "$pid"; then
    if kill -0 "$pid" 2>/dev/null; then
      :  # the server still runs
    else
      # The server may have exited between the last pid-file check and
now.
      if test -n "$avoid_race_condition"; then
        avoid_race_condition=""
        continue  # Check again.
      fi

      # there's nothing that will affect the file.
      log_failure_msg "The server quit without updating PID file
($pid_file_path)."
      return 1  # not waiting any more.
    fi
  fi

  echo $echo_n ".$echo_c"
  i=`expr $i + 1`
  sleep 1

done

if test -z "$i" ; then
  log_success_msg
  return 0
else
  log_failure_msg
  return 1
fi
}

# Get arguments from the my.cnf file,
```

```
# the only group, which is read from now on is [mysqld]
if test -x ./bin/my_print_defaults
then
  print_defaults="./bin/my_print_defaults"
elif test -x $bindir/my_print_defaults
then
  print_defaults="$bindir/my_print_defaults"
elif test -x $bindir/mysql_print_defaults
then
  print_defaults="$bindir/mysql_print_defaults"
else
  # Try to find basedir in /etc/my.cnf
  conf=/etc/my.cnf
  print_defaults=
  if test -r $conf
  then
    subpat='^[^=]*basedir[^=]*=(.*)$'
    dirs=`sed -e "/$subpat/!d" -e 's//1/' $conf`
    for d in $dirs
    do
      d=`echo $d | sed -e 's/[         ]//g'`
      if test -x "$d/bin/my_print_defaults"
      then
        print_defaults="$d/bin/my_print_defaults"
        break
      fi
      if test -x "$d/bin/mysql_print_defaults"
      then
        print_defaults="$d/bin/mysql_print_defaults"
        break
      fi
    done
  fi

  # Hope it's in the PATH ... but I doubt it
  test -z "$print_defaults" && print_defaults="my_print_defaults"
fi

#
# Read defaults file from 'basedir'.   If there is no defaults file there
# check if it's in the old (depricated) place (datadir) and read it from there
#
```

```
extra_args=""
if test -r "$basedir/my.cnf"
then
  extra_args="-e $basedir/my.cnf"
else
  if test -r "$datadir/my.cnf"
  then
    extra_args="-e $datadir/my.cnf"
  fi
fi

parse_server_arguments `$print_defaults $extra_args mysqld server
mysql_server mysql.server`

#
# Set pid file if not given
#
if test -z "$mysqld_pid_file_path"
then
# bug fix
#   mysqld_pid_file_path=$datadir/`hostname`.pid
  mysqld_pid_file_path=$datadir/`/bin/hostname`.pid
else
  case "$mysqld_pid_file_path" in
   /* ) ;;
    * )  mysqld_pid_file_path="$datadir/$mysqld_pid_file_path" ;;
  esac
fi

case "$mode" in
  'start')
    # Start daemon

    # Safeguard (relative paths, core dumps..)
    cd $basedir

    echo $echo_n "Starting MySQL"
    if test -x $bindir/mysqld_safe
    then
      # Give extra arguments to mysqld with the my.cnf file. This script
      # may be overwritten at next upgrade.
      $bindir/mysqld_safe --datadir="$datadir" --pid-
file="$mysqld_pid_file_path" $other_args >/dev/null 2>&1 &
```

```
      wait_for_pid created "$!" "$mysqld_pid_file_path"; return_value=$?

     # Make lock for RedHat / SuSE
     if test -w "$lockdir"
     then
       touch "$lock_file_path"
     fi

     exit $return_value
   else
     log_failure_msg "Couldn't find MySQL server ($bindir/mysqld_safe)"
   fi
   ;;

 'stop')
    # Stop daemon. We use a signal here to avoid having to know the
    # root password.

   if test -s "$mysqld_pid_file_path"
   then
     mysqld_pid=`cat "$mysqld_pid_file_path"`

    if (kill -0 $mysqld_pid 2>/dev/null)
     then
        echo $echo_n "Shutting down MySQL"
       kill $mysqld_pid
        # mysqld should remove the pid file when it exits, so wait for it.
        wait_for_pid removed "$mysqld_pid" "$mysqld_pid_file_path";
return_value=$?
     else
       log_failure_msg "MySQL server process #$mysqld_pid is not
running!"
       rm "$mysqld_pid_file_path"
    fi

     # Delete lock for RedHat / SuSE
     if test -f "$lock_file_path"
     then
       rm -f "$lock_file_path"
    fi
     exit $return_value
   else
     log_failure_msg "MySQL server PID file could not be found!"
```

```
    fi
    ;;

  'restart')
     # Stop the service and regardless of whether it was
    # running or not, start it again.
    if $0 stop  $other_args; then
      $0 start $other_args
    else
      log_failure_msg "Failed to stop running server, so refusing to try to
start."
     exit 1
    fi
    ;;

  'reload'|'force-reload')
    if test -s "$mysqld_pid_file_path" ; then
      read mysqld_pid <  "$mysqld_pid_file_path"
      kill -HUP $mysqld_pid && log_success_msg "Reloading service MySQL"
      touch "$mysqld_pid_file_path"
    else
      log_failure_msg "MySQL PID file could not be found!"
     exit 1
   fi
   ;;
  'status')
    # First, check to see if pid file exists
    if test -s "$mysqld_pid_file_path" ; then
      read mysqld_pid < "$mysqld_pid_file_path"
     if kill -0 $mysqld_pid 2>/dev/null ; then
        log_success_msg "MySQL running ($mysqld_pid)"
       exit 0
      else
        log_failure_msg "MySQL is not running, but PID file exists"
       exit 1
     fi
    else
      # Try to find appropriate mysqld process
      mysqld_pid=`pidof $libexecdir/mysqld`
      if test -z $mysqld_pid ; then
        if test -f "$lock_file_path" ; then
          log_failure_msg "MySQL is not running, but lock file
($lock_file_path) exists"
```

```
        exit 2
      fi
       log_failure_msg "MySQL is not running"
      exit 3
    else
      log_failure_msg "MySQL is running but PID file could not be found"
      exit 4
    fi
  fi
  ;;
  *)
    # usage
    basename=`basename "$0"`
    echo "Usage: $basename  {start|stop|restart|reload|force-
reload|status}  [ MySQL server options ]"
    exit 1
  ;;
esac

exit 0
```