Running Multiple MySQL Instances on one server in CentOS 7/RHEL 7/Fedora

To install multiple instance of MySQL on single server we may consider the following two techniques.

1. Using mysqld_multi with group my.cnf configuration file provided by MySQL itself.
2. Installing manual separate service for multiple instances with separate my.cnf configuration file.

Installing Multiple Instance of MySQL with separate Configuration file (my.cnf)

To run multiple instances using MySQL we need to have a couple of things separate from the initial install on MySQL like data directory, init script and config file.

I will install 4 Instances on ports 3307,3308,3309,3310. Our structure for all instances as below

Port: 3307
DataDir: /var/lib/mysql3307
Config: /etc/my3307.cnf
Service name: /etc/init.d/mysql3307

Port: 3308
DataDir: /var/lib/mysql3308
Config: /etc/my3308.cnf
Service name: /etc/init.d/mysql3308

Port: 3309
DataDir: /var/lib/mysql3309
Config: /etc/my3309.cnf
Service name: /etc/init.d/mysql3309

Port: 3310
DataDir: /var/lib/mysql3310
Config: /etc/my3310.cnf
Service name: /etc/init.d/mysql3310

Following steps will explain how to achieve the same...

I will show the steps for first instance Port: 3307 , Once successful then follow the same steps for other ports.

1. **Create a new data directory [/var/lib/mysql2] and make mysql user own it.**

```
mkdir /var/lib/mysql3307
chown -R mysql.mysql /var/lib/mysql3307/
```

2. **Create / copy existing mysql configuration file, call it my3307.cnf and update data directory/port values and socket path as shown above.**

```
cp /etc/my.cnf /etc/my3307.cnf
vi /etc/my3307.cnf

[mysqld]
datadir=/var/lib/mysql3307
socket=/var/lib/mysql3307/mysqld.sock
server-id=5 ##as per your need
port=3307
```

```
[mysqld_safe]
log-error=/var/log/mysqld3307.log
```

3. **Find my service script shown in the end of this document and search (ctrl+f) some string and replace the same with corresponding 3307 port configuration.**
   - Find **conf=/etc/my3307.cnf** and replace with corresponding new cnf
   - Find **-c/etc/my3307.cnf** and replace with corresponding new cnf
   - Find **$bindir/mysqld_safe --defaults-file=/etc/my3310.cnf** and replace with corresponding new cnf
   - Save new modified service script as **/etc/init.d/mysql3307**
   - **Chmod –R 777 /etc/init.d/mysql3307**

4. **Copy any existing datadirectory or Install default tables for this new database instance**

```
Cp –R /var/lib/mysql /var/lib/mysql33017
or
mysql_install_db --datadir=/var/lib/mysql3307 --defaults-file=/etc/my3307.cnf --user=mysql
```

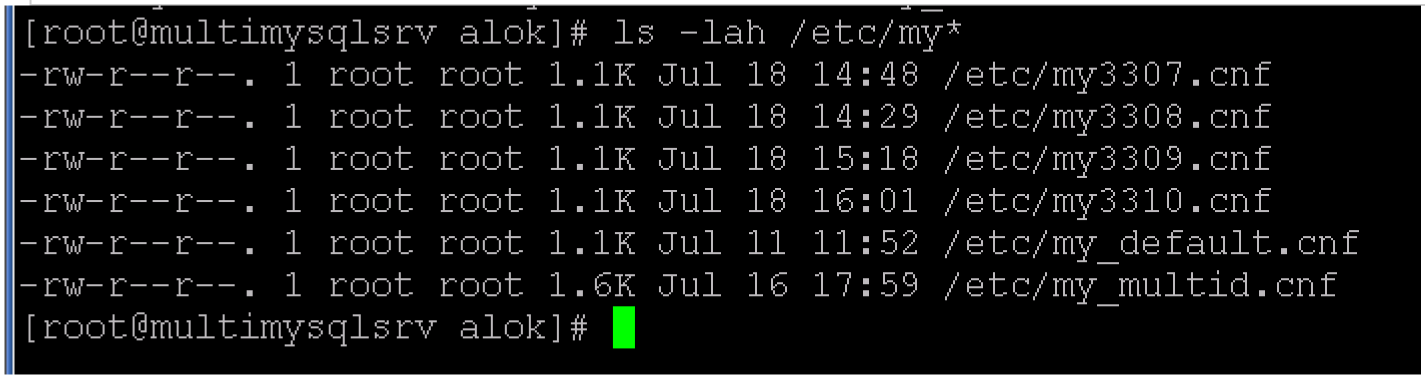
5. **Start the new instance**

```
Service mysql3307 start
```

6. **Follow the steps from 1-5 for all other instance 3308/3309/3310.**

Verifying/testing:

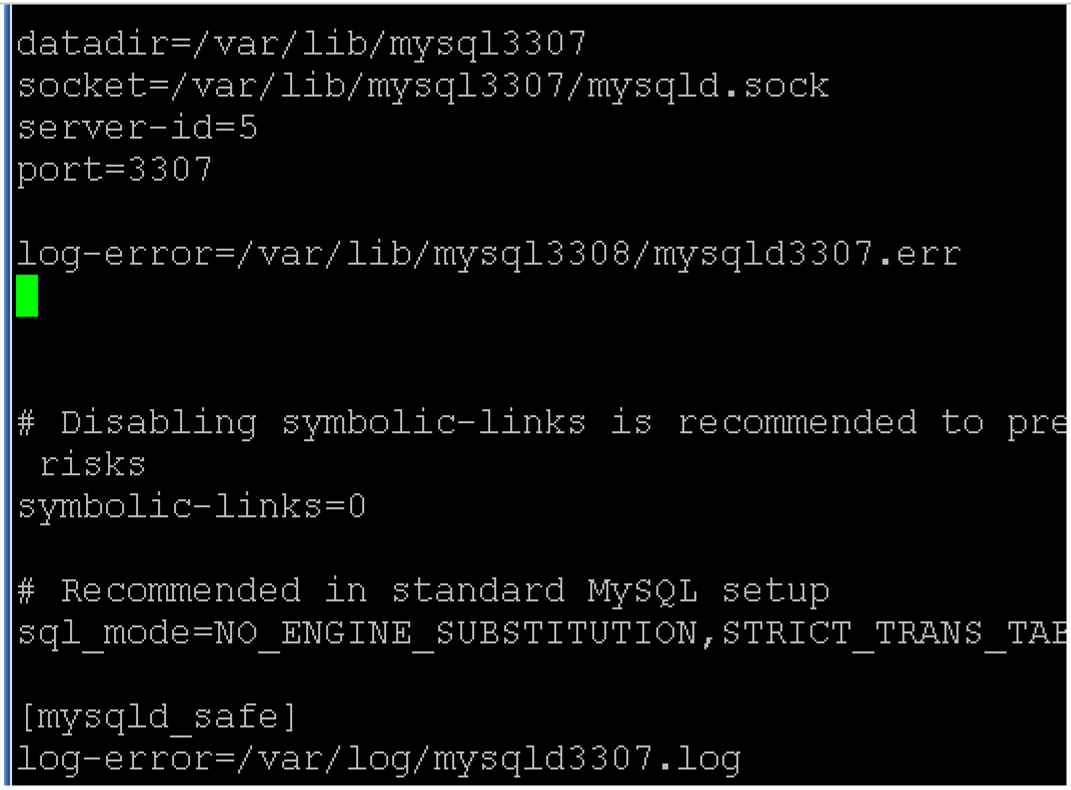
```
Ls /etc/my*
```



```
[root@multimysqlsrv alok]# ls -lah /etc/my*
-rw-r--r--. 1 root root 1.1K Jul 18 14:48 /etc/my3307.cnf
-rw-r--r--. 1 root root 1.1K Jul 18 14:29 /etc/my3308.cnf
-rw-r--r--. 1 root root 1.1K Jul 18 15:18 /etc/my3309.cnf
-rw-r--r--. 1 root root 1.1K Jul 18 16:01 /etc/my3310.cnf
-rw-r--r--. 1 root root 1.1K Jul 11 11:52 /etc/my_default.cnf
-rw-r--r--. 1 root root 1.6K Jul 16 17:59 /etc/my_multid.cnf
[root@multimysqlsrv alok]#
```

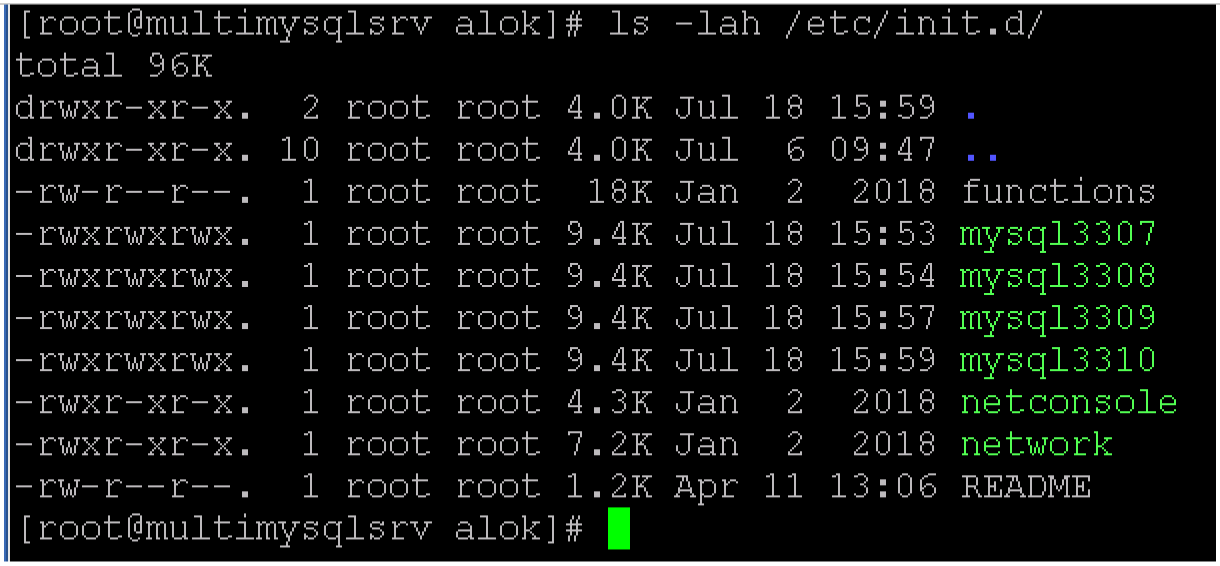
Running Multiple MySQL Instances on one server in CentOS 7/RHEL 7/Fedora

```
Vi /etc/my3307.cnf
```



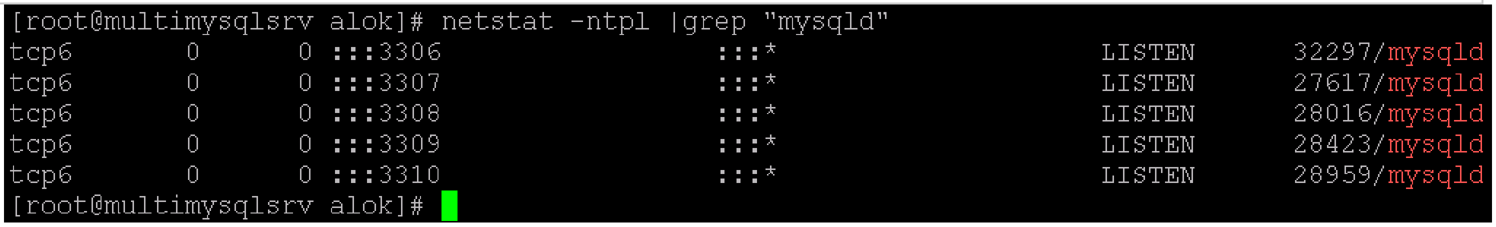
```
datadir=/var/lib/mysql3307
socket=/var/lib/mysql3307/mysqld.sock
server-id=5
port=3307

log-error=/var/lib/mysql3308/mysqld3307.err



# Disabling symbolic-links is recommended to pre
 risks
symbolic-links=0

# Recommended in standard MySQL setup
sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TAE

[mysqld_safe]
log-error=/var/log/mysqld3307.log
```

```
Ls -lah /etc/init.d/mysql3307
```



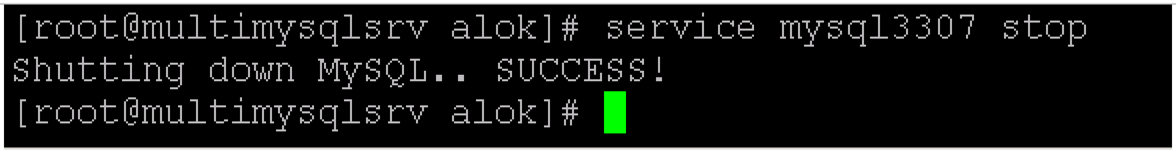
```
[root@multimysqlsrv alok]# ls -lah /etc/init.d/
total 96K
drwxr-xr-x.  2 root root 4.0K Jul 18 15:59 .
drwxr-xr-x. 10 root root 4.0K Jul  6 09:47 ..
-rw-r--r--.  1 root root  18K Jan  2  2018 functions
-rwxrwxrwx.  1 root root 9.4K Jul 18 15:53 mysql3307
-rwxrwxrwx.  1 root root 9.4K Jul 18 15:54 mysql3308
-rwxrwxrwx.  1 root root 9.4K Jul 18 15:57 mysql3309
-rwxrwxrwx.  1 root root 9.4K Jul 18 15:59 mysql3310
-rwxr-xr-x.  1 root root 4.3K Jan  2  2018 netconsole
-rwxr-xr-x.  1 root root 7.2K Jan  2  2018 network
-rw-r--r--.  1 root root 1.2K Apr 11 13:06 README
[root@multimysqlsrv alok]#
```

```
Netstat –ntpl | grep “mysqld”
```



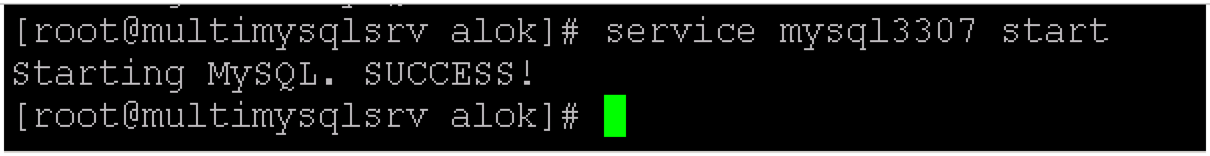
```
[root@multimysqlsrv alok]# netstat -ntpl |grep "mysqld"
tcp6       0      0 :::3306                 :::*                    LISTEN      32297/mysqld
tcp6       0      0 :::3307                 :::*                    LISTEN      27617/mysqld
tcp6       0      0 :::3308                 :::*                    LISTEN      28016/mysqld
tcp6       0      0 :::3309                 :::*                    LISTEN      28423/mysqld
tcp6       0      0 :::3310                 :::*                    LISTEN      28959/mysqld
[root@multimysqlsrv alok]#
```

```
Service mysql3307 stop
```



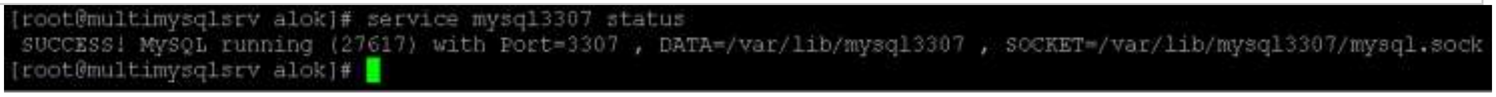
```
[root@multimysqlsrv alok]# service mysql3307 stop
Shutting down MySQL.. SUCCESS!
[root@multimysqlsrv alok]#
```

```
Service mysql3307 start
```



```
[root@multimysqlsrv alok]# service mysql3307 start
Starting MySQL. SUCCESS!
[root@multimysqlsrv alok]#
```

```
Service mysql3307 status
```



```
[root@multimysqlsrv alok]# service mysql3307 status
 SUCCESS! MySQL running (27617) with Port=3307 , DATA=/var/lib/mysql3307 , SOCKET=/var/lib/mysql3307/mysql.sock
[root@multimysqlsrv alok]#
```

Please find my final mysqld service script for instance 3307 (/etc/init.d/mysql3307)

https://github.com/getmysql/MySQLMultiInstance/blob/master/mysql3307

```
#!/bin/sh
# Copyright Alok Kumar Singh @www.getmysql.info
# MySQL daemon start/stop script.

# Usually this is put in /etc/init.d/[new service name]
# Provide chmod -R 777 to [new service name]

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 "$basedir"
then
  basedir=/usr
  bindir=/usr/bin
  if test -z "$datadir"
  then
    datadir=/var/lib/mysql
  fi
  sbindir=/usr/sbin
  libexecdir=/usr/sbin
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 "testing\c"`,`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 "$arg" | sed -e 's/^[^=]*=//'` ;;
      --port=*)     port=`echo "$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" > /dev/null 2> /dev/null
     #if kill -0 $mysqld_pid > /dev/null 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/my3307.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 -
c/etc/my3307.cnf`

#
# Set pid file if not given
#
if test -z "$mysqld_pid_file_path"
then
  mysqld_pid_file_path=/var/run/mysqld/mysqld$port.pid
  #mysqld_pid_file_path=$pidfile
else
  case "$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.
      $bindir/mysqld_safe --defaults-file=/etc/my3307.cnf --datadir="$datadir" --pid-
file="$mysqld_pid_file_path" --port="$port" --socket="$datadir"/mysql.sock $other_args >/dev/null
2>&1 &
      wait_for_pid created "$!" "$mysqld_pid_file_path"; return_value=$?

      # 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 -gt /dev/null)
       if kill -0 $mysqld_pid > /dev/null 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 "$mysqld_pid_file_path" ; then
     mysqld_pid=$(cat "$mysqld_pid_file_path")
        #read mysqld_pid &lt; "$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 "$mysqld_pid_file_path" ; then
        mysqld_pid=`cat "$mysqld_pid_file_path"`
       #read mysqld_pid &lt; "$mysqld_pid_file_path"
        #if kill -0 $mysqld_pid 2 -gt /dev/null ;
       if kill -0 $mysqld_pid > /dev/null 2> /dev/null
       then
          #log_success_msg "MySQL running ($mysqld_pid)"
            log_success_msg "MySQL running ($mysqld_pid) with Port=$port , DATA=$datadir ,
  SOCKET=$datadir/mysql.sock "
          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 not running"
          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
```