Qemu-Geust-Agent功能验证&学

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 修订者 | 修订描述 |
| 2017.11.09 | v1 | 林 清 | 初稿 |
| 2017.11.10 | v1.1 | 林 清 |  |
| 2017.11.13 | v1.2 | 林 清 |  |
|  |  |  |  |

[1. 介绍 2](#_Toc498364129)

[1.1 Qemu-ga通信原理 2](#_Toc498364130)

[1.2 消息通道 2](#_Toc498364131)

[1.3 配置qemu-guest-agent 3](#_Toc498364132)

[1.3.1 Linux Guest 3](#_Toc498364133)

[1.3.2 Windows Guest 3](#_Toc498364134)

[1.4 在宿主机上进行接口调用 4](#_Toc498364135)

[1.4.1 通过virsh 4](#_Toc498364136)

[1.4.2 通过socat 4](#_Toc498364137)

[1.5 安装编译 5](#_Toc498364138)

[1.5.1 编译安装Linux环境下的Qemu-ga 5](#_Toc498364139)

[1.5.2 编译安装Windows环境下的Qemu-ga 5](#_Toc498364140)

[1.6 定制Qemu-guest-agent功能接口 6](#_Toc498364141)

[1.6.1 示例：Win32平台的内存利用率 6](#_Toc498364142)

[1.7 Windows平台下附加功能的开发 7](#_Toc498364143)

[1.7.1 统计CPU使用率 7](#_Toc498364144)

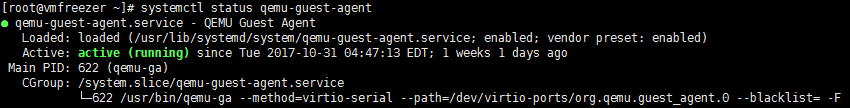
[1.7.2 统计内存使用率 7](#_Toc498364145)

[1.7.3 统计磁盘使用率 7](#_Toc498364146)

# 介绍

## Qemu-ga通信原理

qga是一个运行在虚拟机内部的普通应用程序，可执行文件名称默认为qemu-ga，服务名称默认为qemu-guest-agent。



qga实现一种宿主机和虚拟机进行交互的方式，这种方式不依赖于网络，而是依赖于virtio-serial（默认首选方式）或者isa-serial，而QEMU则提供了串口设备的模拟及数据交换的通道，最终呈现出来的是一个串口设备（虚拟机内部）和一个unix socket文件（宿主机上）。

虚机内部的串口设备：



宿主机中定义的unix socket文件：



<channel type='unix'>

<source mode='bind' path='/var/lib/libvirt/qemu/org.qemu.guest\_agent.0.instance-0000ccb9.sock'/>

<target type='virtio' name='org.qemu.guest\_agent.0' state='connected'/>

<alias name='channel0'/>

<address type='virtio-serial' controller='0' bus='0' port='1'/>

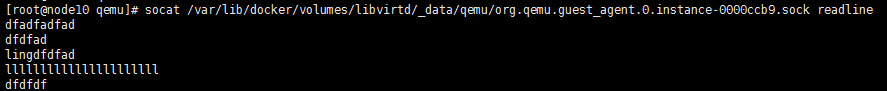
</channel>

qga通过读写串口设备与宿主机上的socket通道进行交互，宿主机上可以使用普通的unix socket读写方式对socket文件进行读写，最终实现与qga的交互。交互的协议与qmp（QEMU Monitor Protocol）相同，串口设备的速率通常都较低，所以比较适合小数据量的交换。

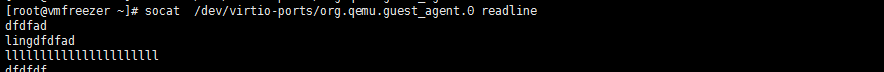
## 消息通道

通过socat工具可以在宿主机上直接通过unix sock直接和虚机通信。安装socat工具之后，可以在宿主机和虚拟机之间进行消息收发。

在宿主机中：



在虚拟机中：



## 配置qemu-guest-agent

### Linux Guest

登录Guest系统，通过qemu-ga命令或者配置文件/etc/qemu-ga/qemu-ga.conf。

值得关注的是配置项fsfreeze-hook，通过该配置项可以让虚拟机文件系统冻结前后执行一系列用户定制的脚本。脚本存放在/etc/qemu-ga/fsfreeze-hook.d目录。

配置文件示例：

[general]

verbose = 0

method = virtio-serial

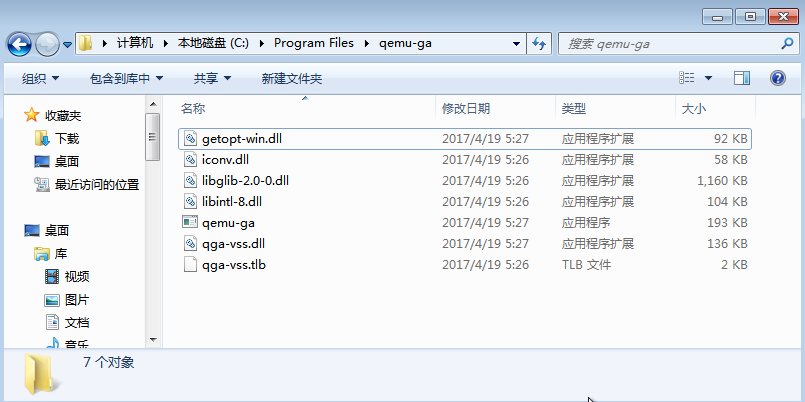
logfile = /var/log/qemu-ga/qemu-ga.log

fsfreeze-hook = /etc/qemu-ga/fsfreeze-hook

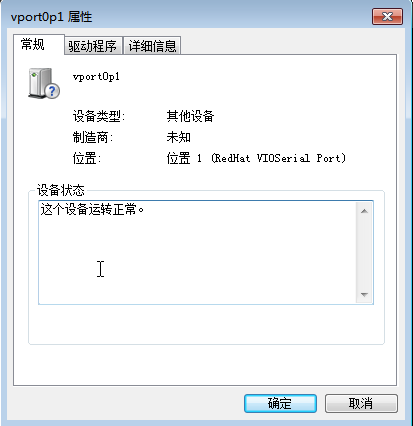
### Windows Guest

在Windows中安装了qga之后，可以看到对应的QEMU Guest Agent服务和QEMU Guest Agent VSS Provider服务。

对应的qemu-ga在目录C://Program Files/qemu-ga，通过cmd能够配置qga服务的日志文件等参数。



在驱动设备中可以看到QEMU虚拟出的设备（注意，需要右击设备管理器->查看->显示隐藏设备）:



## 在宿主机上进行接口调用

### 通过virsh

在宿主机上进入nova\_libvirt容器通过virsh的子命令qemu-agent-command调用虚机中的qga服务。

# 获取虚拟机中qga的版本信息

virsh qemu-agent-command <instance-id> {"execute":"guest-info" }

### 通过socat

使用socat可以直接向虚拟机中发送数据。

# 安装socat

yum –y install socat

# 建立socat连接

socat /var/lib/docker/volumes/libvirtd/\_data/qemu/org.qemu.guest\_agent.0.instance-0000ccb9.sock readline

需要注意的是sock连接不支持多个进程向一个管道同时发送消息，所以通过socat建立管道之后，需要重启虚机的qga服务。



## 安装编译

qemu-guest-agent不是一个独立的项目，而是属于qemu的一部分。因此如果从源码安装qemu-guest-agent则需要获取整个qemu项目的源码。

当前qemu的稳定版本branch是stable-2.10。

### 编译安装Linux环境下的Qemu-ga

git clone <https://github.com/qemu/qemu.git>

yum install gcc zlib-devel.x86\_64 glib2-devel.x86\_64 pixman-devel.x86\_64

git submodule update --init dtc

mkdir build

cd build

../configure --enable-guest-agent --target-list=arm-linux-user --datadir=/var --bindir=/usr/bin --libdir=/usr/lib --sysconfdir=/etc --localstatedir=/var

make qemu-ga && make install qemu-ga

# 启动守护进程

qemu-ga –d

### 编译安装Windows环境下的qemu-ga.msi

编译qemu-ga.msi时，我们可以通过x86\_64-w64-mingw32-gcc在Linux环境上使用编译。

同时由于需要打包msi，因此要在Linux环境上安装msitools。

# 获取安装包

wget <http://ftp.gnome.org/pub/GNOME/sources/msitools/0.97/msitools-0.97.tar.xz>

tar -xvJf msitools-0.97.tar.xz

# 创建build目录

cd msitools-0.97

mkdir build

cd build

# 安装编译依赖

yum -y install gcc libgsf-devel libuuid-devel libgcab1-devel

# configure

../configure

# 编译安装

make . && make install

这里需要注意，使用x86\_64-w64-mingw32-gcc需要安装第三方的pthreads库。

# 获取安装包

wget http://wiki.qemu-project.org/download/qemu-2.10.1.tar.bz2

# 解压

bunzip2 qemu-2.10.1.tar.bz2

tar -xf qemu-2.10.1.tar

# 安装mingw32工具

yum install -y mingw64-pixman mingw64-glib2 mingw64-gmp mingw64-SDL mingw64-pkg-config

# 安装pthread

yum -y install mingw32-winpthreads-static mingw64-winpthreads-static mingw32-winpthreads mingw64-winpthreads

# 创建build目录

mkdir build

cd build

# 配置

../configure --enable-guest-agent --enable-guest-agent-msi --cross-prefix=x86\_64-w64-mingw32-

# 编译

make qemu-ga

## 定制Qemu-guest-agent功能接口

定制qga接口需要修改以下三个文件：

# 定义结构体，以及声明名自定义接口

qga/qapi-schema.json

# 自定义接口的Linux实现

qga/commands-posix.c

# 自定义接口的Win32实现

qga/commands-win32.c

与需要注意几点：

1. qapi-schema.json中定义了接口和结构体后，编译时会自动在build/qga/qapi-generated目录生成一系列的头文件。
2. 一般情形下不允许返回原生类型，自定义接口需要返回qapi-schema.json中定义的结构体。
3. 需要同时实现posix和win32中的接口，不可只实现一个（当然可以在实现中直接抛异常）
4. qapi-schema.json中定义命令XXX-XXX，那么在.c文件中对应的实现是qmp\_XXX\_XXX

### 示例：Win32平台的内存利用率

# 定义结构体，以及声明名自定义接口

##

# @GuestMemInfo:

#

# Information about Memory usage.

#

# @total: total size of Memory

#

# @usage: Memory usage

#

# Since 2.6

##

{ 'struct': 'GuestMemInfo',

'data': { 'total': 'int', 'usage': 'int' } }

##

# @guest-get-mem-usage:

#

# Get the memory utilization rate.

#

# Returns: @GuestMemInfo on success.

#

# Since 2.6

##

{ 'command': 'guest-get-mem-usage',

'returns': 'GuestMemInfo' }

# Win32实现

GuestMemInfo \*qmp\_guest\_get\_mem\_usage(Error \*\*errp)

{

MEMORYSTATUS ms;

GuestMemInfo \*meminfo = g\_new0(GuestMemInfo, 1);

GlobalMemoryStatus(&ms);

meminfo->usage = (int64\_t)ms.dwMemoryLoad;

meminfo->total = (int64\_t)ms.dwTotalPhys;

return meminfo;

}

# POSIX 不实现直接抛出异常

GuestMemInfo \*qmp\_guest\_get\_mem\_usage(Error \*\*errp)

{

error\_setg(errp, QERR\_UNSUPPORTED);

}

## Windows平台下附加功能的开发

### 统计CPU使用率

通过GetSystemTimes函数,可以获取操作系统某个时间段内,所有CPU的系统时间、用户时间、以及空闲时间总和。

CPU使用率计算公式：

CPU% = abs((kernelTime + userTime - idleTime) / (kernelTime + userTime))\*100%

### 统计内存使用率

通过GlobalMemoryStatusEx可以获取当前物理内存、虚拟内存使用率。

参考：<https://msdn.microsoft.com/en-us/library/aa366589(VS.85).aspx>

### 统计磁盘使用率

通过下面的 Windows API能够获取CPU使用率。

GetLogicalDrives、GetLogicalDriveStrings、GetDriveType、GetDiskFreeSpaceEx

## FAQ

### 指定QGA的版本号

QGA版本号即为QEMU版本号，格式为：

QEMU\_VERSION\_MAJOR. QEMU\_VERSION\_MINOR. QEMU\_VERSION\_MICRO

编译时通过修改VERSION文件能够指定版本号，我们可以将版本号拓展到4四位，或者在VERSION\_MICRO上加上100作为偏移。比如基于2.10.1修改的版本我们可以以2.10.1XX作为我们的版本号方便升级管理。

### 在Windows环境中安装Virtio-serial驱动

部分Windows镜像中Virtio-serial的驱动没有预先安装分，会导致GEMU Guest Agent服务无法启动。需要手工安装Virtio-serial驱动。

### Windows环境中的Service管理

创建Service

sc create QEMU-GA binPath= "C:\Program Files\qemu-ga\qemu-ga.exe -d” start= "auto" DisplayName= "QEMU Guest Agent"

删除Service

sc delete QEMU-GA

### 关于Windows版本时的版本号

通过qemu-ga.msi安装时，预装版本的Qemu-ga没有使用的qemu的版本号，而是使用独立的版本号，版本是7.3.2。因此如果直接使用自己编译的安装包，安装时可能会出现版本过低无法覆盖安装的情况。