SPICE源码分析

一、 spice server

qemu通过spice\_server\_new创建一个reds对象，将该reds传给spice\_server\_init进行初始化，具体的初始化过程在do\_spice\_init(core)中完成。core是qemu中的一个结构对象。

reds = spice\_new0(RedsState, 1);

spice\_server\_init(reds, core)

do\_spice\_init(core);

main\_dispatcher\_init(core); //初始化main\_dispatcher

reds\_init\_net()

main\_channel\_init();

inputs\_init();

main\_dispatcher\_init实现main\_dispatcher的初始化，将main\_dispatcher的recv\_fd添加到core的监听队列中，由qemu监听该文件描述符，并执行dispatcher\_handle\_read回调函数。最后注册3个消息处理函数。

main\_dispatcher\_init()

dispatcher\_init()

core->watch\_add()

dispatcher\_register\_handler()

网络secket的初始化，并把相应的socket添加到qemu的监听队列中。

reds\_init\_net()

reds\_init\_socket() //创建socket

core->watch\_add() //添加对socket的监听，回调函数reds\_accept()

---------------------------------------------------------------------------------------------------------------------------------

reds\_accept()

socket = accept()

spice\_server\_add\_client(reds, socket, 0) //添加一个client

添加新的client连接

spice\_server\_add\_client()

reds\_init\_client\_connection(socket)

reds\_handle\_new\_link(link);

---------------------------------------------------------------------------------------------------------------------------------

reds\_init\_client\_connection(socket)

//创建RedsStream \*stream; 并进行初始化

reds\_stream\_push\_channel\_event(stream, SPICE\_CHANNEL\_EVENT\_CONNECTED);

main\_dispatcher\_channel\_event(event, s->info);

dispatcher\_send\_message()

二、 spicec