Unclassified description of FB:

Introspection of cyber activity using a hypervisor (computer virtualization)

Clarification:

* obviously, only can describe the unclassified description of FB
* everything else (such as the open source aspects of QEMU/KVM) is open source and allowed to be discussed, correct?

FB Learning

Writing QMP commands using the QAPI framework

General Steps Taken

1. Write the command and type’s specification in the QAPI schema file (the qapi-schema.json in the root source directory)
2. Write the QMP command in C; preferably the command should be exported by some QEMU subsystem (ie, frost\_qmp.c instead of qmp.c)
3. Test the command under the QMP protocol
4. Write the HMP command equivalent; though this is not required, and only should be done if it makes sense to have the functionality in HMP; the HMP command is implemented in terms of the QMP command

**Testing 1 (basics):**

* QEMU is started as…

$ sudo ./i386-softmmu/qemu-system-i386 -chardev socket,id=qmp,port=4444,host=localhost,server -mon chardev=qmp,mode=control,pretty=on -monitor stdio -m 1024 -hda ~/vms/xpsp2-x86.img -net none -machine accel=kvm

* In a different terminal…

$ telnet localhost 4444

* Then should get the following shown on the terminal which basically means the QMP server is saying you’re connected
  + INSERT IMAGE
* Next, need to enter in command mode type

{ “execute” : “qmp\_capabilities” }

* The server responds with the following to basically say the latest command was executed OK and didn’t return any data…
  + INSERT IMAGE
* There are a bunch of query commands you can access such as system\_reset
  + INSERT IMAGE OF QUERY COMMANDS
* System\_reset example
  + INSERT IMAGE OF WINDOWS XP VM AND THE QEMU TERMINAL

**Testing 2 (hello-world):**

* Now I implement a simple command that doesn’t return any data
* Print hello world to the standard output
* Edit the json file to define my new QMP command, a JSON object
  + { ‘command’ : ‘hello-world’ }
  + the QAPI will automatically generate any prototypes and the necessary code to marshal and unmarshal protocol data
* next, write the hello-world implementation
  + void qmp\_hello\_world(Error \*\*errp) { printf(“Hello, world!\n”); }
  + note that…
    - the QMP command implementation functions are prefixed with “qmp\_”
    - it takes an “Error \*\*” argument; this is required; the Error argument should not be touched if the command doesn’t return errors
    - the function’s prototype is automatically added by the QAPI
    - normally, printing to the terminal is discouraged, but as a demonstration, it works great
* Editing the internal dispatch table in the qmp-commands.hx file

{

.name = “hello-world”,

.args\_type = “”,

.mhandler.cmd\_new = qmp\_marshal\_input\_hello\_world,

},

* Now, rebuild qemu and run just like earlier
  + Typing in the following command:
    - { “execute” : “hello-world” }
  + yields the output in the other terminal
    - Hello, world!

**Testing 3 (less typing via a python wrapper script)**

**Testing 4 (display packet information using custom QMP command)**

**Testing 5 (displaying packet information from two different terminals and distinguishing the packet information—one from linux and one from mac)**