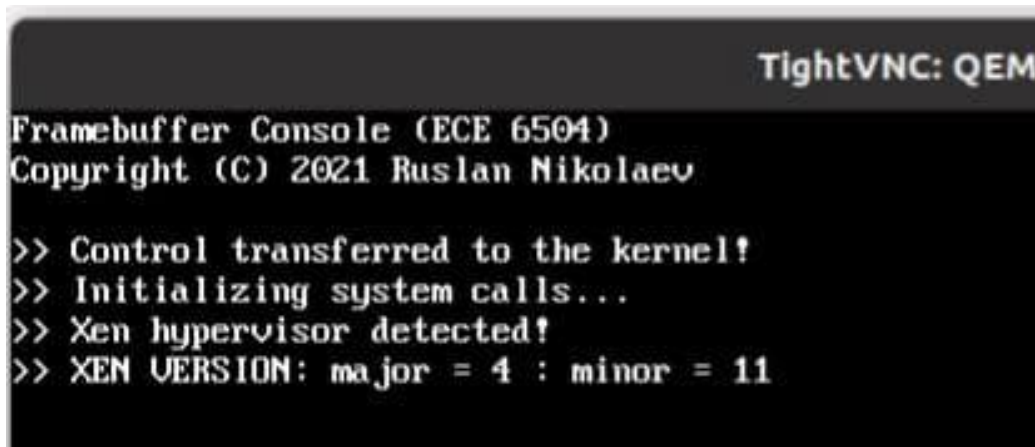


Assignment 3

Carlos Bilbao (bilbao@vt.edu)

2.1 Xen Initialization [25 pts]

I write function `init_xen()` in file `kernel.c` to detect the hypervisor, initialize hypercalls and map the shared info structure. I invoke functions from my new file `basic_xen.c` from it. I use code from `rumprun-smp` but only for the Xen hypervisor (discarding the code for others). I make the invocation of the Xen code conditionally compiled with a variable defined at `basic_xen.h` (that I leave set for convenience). To print the major and minor numbers I obtain the version with the `HYPervisor_xen_version` hypercall, and then obtain the first or last 16 bits with bitwise operations. The major and minor numbers are successfully printed for Xen 4.11:

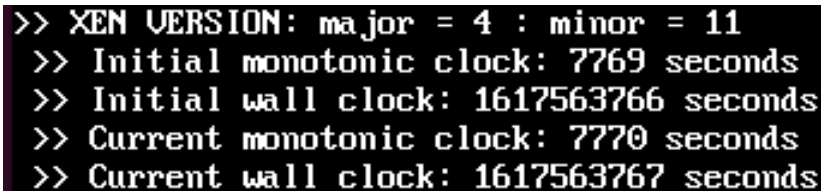
A screenshot of a terminal window titled "TightVNC: QEM". The terminal shows the output of a program. It starts with "Framebuffer Console (ECE 6504)" and "Copyright (C) 2021 Ruslan Nikolaev". Then it shows a series of status messages: ">> Control transferred to the kernel!", ">> Initializing system calls...", ">> Xen hypervisor detected!", and ">> XEN VERSION: major = 4 : minor = 11".

```
TightVNC: QEM
Framebuffer Console (ECE 6504)
Copyright (C) 2021 Ruslan Nikolaev

>> Control transferred to the kernel!
>> Initializing system calls...
>> Xen hypervisor detected!
>> XEN VERSION: major = 4 : minor = 11
```

2.2 PV Clock [25 pts]

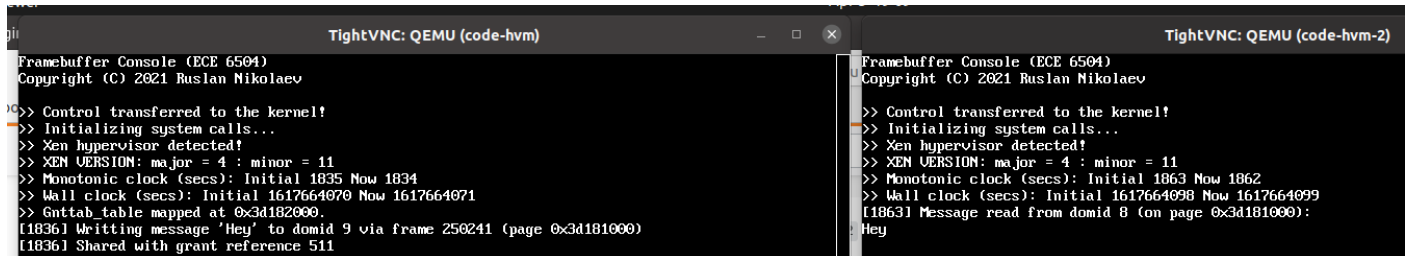
I write function `xen_pv_clock()` in file `kernel.c`, that calls functions for both the monotonic and the wall clock, which retrieve the time in seconds rounding up. To print the wall clock I add the monotonic one and the retrieved offset:

A screenshot of a terminal window showing the output of the `xen_pv_clock()` function. It displays five lines of status messages: ">> XEN VERSION: major = 4 : minor = 11", ">> Initial monotonic clock: 7769 seconds", ">> Initial wall clock: 1617563766 seconds", ">> Current monotonic clock: 7770 seconds", and ">> Current wall clock: 1617563767 seconds".

```
>> XEN VERSION: major = 4 : minor = 11
>> Initial monotonic clock: 7769 seconds
>> Initial wall clock: 1617563766 seconds
>> Current monotonic clock: 7770 seconds
>> Current wall clock: 1617563767 seconds
```

3.1 Shared Memory [25 pts]

I write function `shared_memory_xen()` in file `kernel.c`. I prepare the macro `OTHER_SIDE_DOMID` for granting access to the other guest domain. Likewise, the first guest will have the flag `I_SHARE_DATA` raised. We see in the following image where we write on the left and the right side reads:



```
Framebuffer Console (ECE 6504)
Copyright (C) 2021 Ruslan Nikolaev

>> Control transferred to the kernel!
>> Initializing system calls...
>> Xen hypervisor detected!
>> XEN VERSION: major = 4 : minor = 11
>> Monotonic clock (secs): Initial 1835 Now 1834
>> Wall clock (secs): Initial 1617664070 Now 1617664071
>> Gnttab table mapped at 0x3d182000.
[1836] Writing message 'Hey' to domid 9 via frame 250241 (page 0x3d181000)
[1836] Shared with grant reference 511
```

```
Framebuffer Console (ECE 6504)
Copyright (C) 2021 Ruslan Nikolaev

>> Control transferred to the kernel!
>> Initializing system calls...
>> Xen hypervisor detected!
>> XEN VERSION: major = 4 : minor = 11
>> Monotonic clock (secs): Initial 1863 Now 1862
>> Wall clock (secs): Initial 1617664098 Now 1617664099
[1863] Message read from domid 8 (on page 0x3d181000):
Hey
```

In order to run both guests we need to:

1. For the writer, modify the file `kerninc/gnttab.h` to set `I_SHARE_DATA=1` and the domid of the reader, `OTHER_SIDE_DOMID`. Then edit the Makefile, commenting the lines for `make2` and the new configuration file and uncommenting the `make1` and the old configuration file.
2. To execute the reader, modify the file `kerninc/gnttab.h` to set `I_SHARE_DATA=0` and update the `OTHER_SIDE_DOMID`. Then edit the Makefile, uncommenting the lines for `make2` and the new configuration file and commenting the `make1` and the old configuration file.

3.2 Adding a Hypercall [25 pts]

In order to add my hypercall I follow [this](#) commit and add a new hypercall `__HYPERVISOR_assignment_3`. I declare the hypercall, assigning it an identifier, for both HVM and PV, and write the handler in a new C function. I also include the definition in this function in the corresponding header. I add the extra definitions as described in the message (`xen/arch/x86/hypercall.c` and `xen/arch/x86/guest/xen/hypercall_page.S`). I configure as default (make default) and compile to make sure there is no error. I clean and remove generated files before creating the patch `xen_hypercall.patch`.