Lab BSY IO (SOLUTION)

Introduction & Prerequisites

This laboratory is to learn how to:

- Learn about IO-related aspects of your system, hardware and software related.
- Find and understand the basic information reported on interrupts and performance of Input-Output (IO) topics

The following resources and tools are required for this laboratory session:

- Any modern web browser,
- Any modern SSH client application
- OpenStack Horizon dashboard: https://ned.cloudlab.zhaw.ch
- OpenStack account details: please contact the lab assistant in case you already have not received your access credentials.

Important access credentials:

- Username to login with SSH into VMs in ned.cloudlab.zhaw.ch OpenStack cloud from your laptops
  - **`ubuntu`**

Time

The entire session will take 90 minutes.

Task 0 – Setup

In this task you will set up the virtual lab environment ready for you to work with IO. In this task you will extract and understand various basic pieces of information related to IO on a Linux operating system and extract their meaning.

In order to get started, create a VM instance from the standard image with a flavor size of `m1.medium`. Access the VM with SSH using the username `ubuntu`. Once you have SSH'ed into the VM, change your shell to a root shell (`sudo -i`). This will not require you run '`sudo`' in front of every command that requires root/system level privilege[1].

Most of the tools should be installed on the VM, missing ones can be installed using "`apt`".

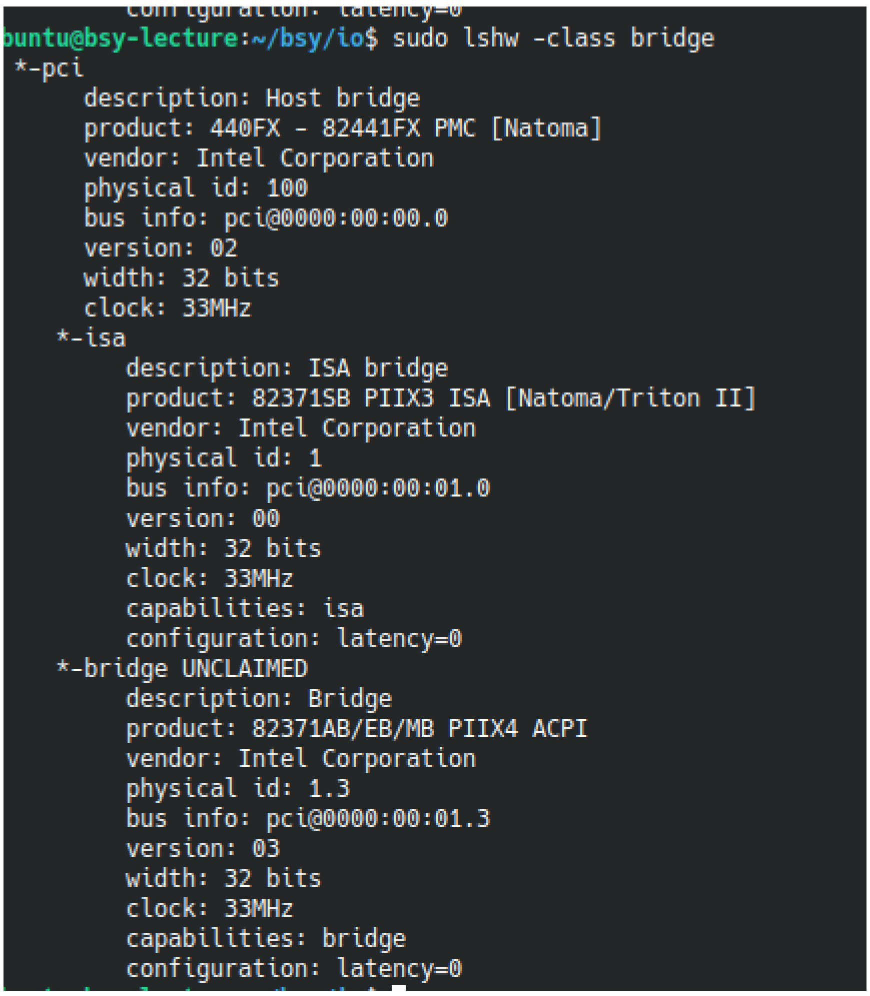
[1] This is for convenience and not recommended to do on production systems

Task 1 – IO Hardware and Architecture

In this task you will carry out some general reconnaissance of the Linux IO subsystem.

Start with getting an overview about the available hardware in your system (hint: `man lshw`).

Get all information about the chipset and identify manufacturers and features.

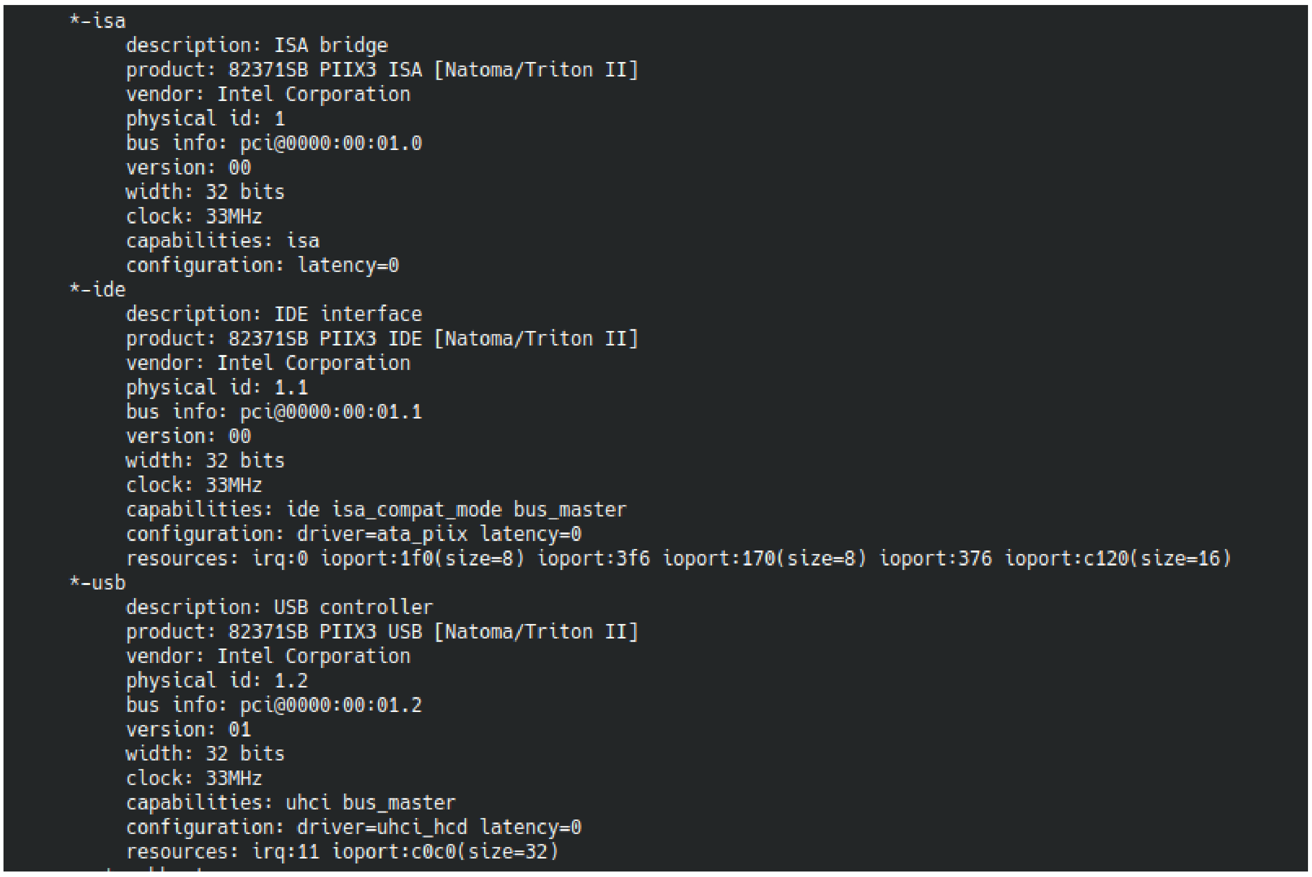


```
buntu@bsy-lecture:~/bsy/io$ sudo lshw -class bridge
 *-pci
      description: Host bridge
      product: 440FX - 82441FX PMC [Natoma]
      vendor: Intel Corporation
      physical id: 100
      bus info: pci@0000:00:00.0
      version: 02
      width: 32 bits
      clock: 33MHz
    *-isa
         description: ISA bridge
         product: 82371SB PIIX3 ISA [Natoma/Triton II]
         vendor: Intel Corporation
         physical id: 1
         bus info: pci@0000:00:01.0
         version: 00
         width: 32 bits
         clock: 33MHz
         capabilities: isa
         configuration: latency=0
    *-bridge UNCLAIMED
         description: Bridge
         product: 82371AB/EB/MB PIIX4 ACPI
         vendor: Intel Corporation
         physical id: 1.3
         bus info: pci@0000:00:01.3
         version: 03
         width: 32 bits
         clock: 33MHz
         capabilities: bridge
         configuration: latency=0
```

Host Bridge (NorthBridge)
https://en.wikipedia.org/wiki/Intel_440FX

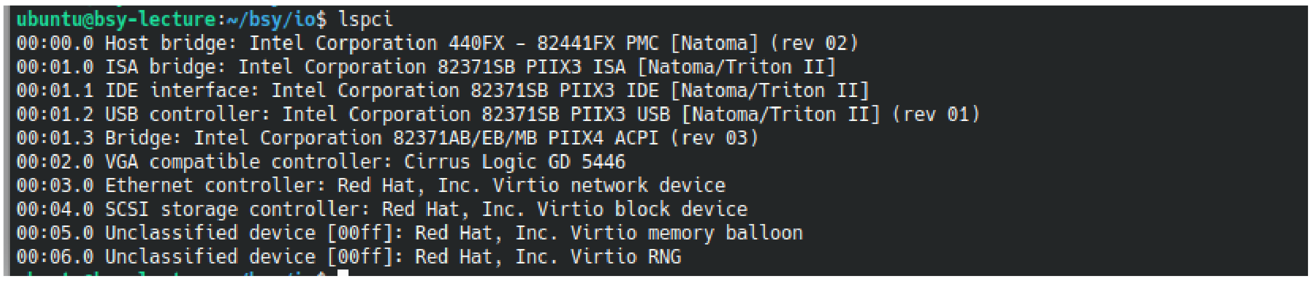
SouthBridge
https://en.wikipedia.org/wiki/PCI_IDE_ISA_Xcelerator#PIIX3
https://en.wikipedia.org/wiki/PCI_IDE_ISA_Xcelerator#PIIX4

The chipset is the central coordinator for the majority of IO related hardware components. Analyze the association of bus systems to chipset components, i.e. for the Northbridge and the Southbridge.



```
     *-isa
          description: ISA bridge
          product: 82371SB PIIX3 ISA [Natoma/Triton II]
          vendor: Intel Corporation
          physical id: 1
          bus info: pci@0000:00:01.0
          version: 00
          width: 32 bits
          clock: 33MHz
          capabilities: isa
          configuration: latency=0
     *-ide
          description: IDE interface
          product: 82371SB PIIX3 IDE [Natoma/Triton II]
          vendor: Intel Corporation
          physical id: 1.1
          bus info: pci@0000:00:01.1
          version: 00
          width: 32 bits
          clock: 33MHz
          capabilities: ide isa_compat_mode bus_master
          configuration: driver=ata_piix latency=0
          resources: irq:0 ioport:1f0(size=8) ioport:3f6 ioport:170(size=8) ioport:376 ioport:c120(size=16)
     *-usb
          description: USB controller
          product: 82371SB PIIX3 USB [Natoma/Triton II]
          vendor: Intel Corporation
          physical id: 1.2
          bus info: pci@0000:00:01.2
          version: 01
          width: 32 bits
          clock: 33MHz
          capabilities: uhci bus_master
          configuration: driver=uhci_hcd latency=0
          resources: irq:11 ioport:c0c0(size=32)
```

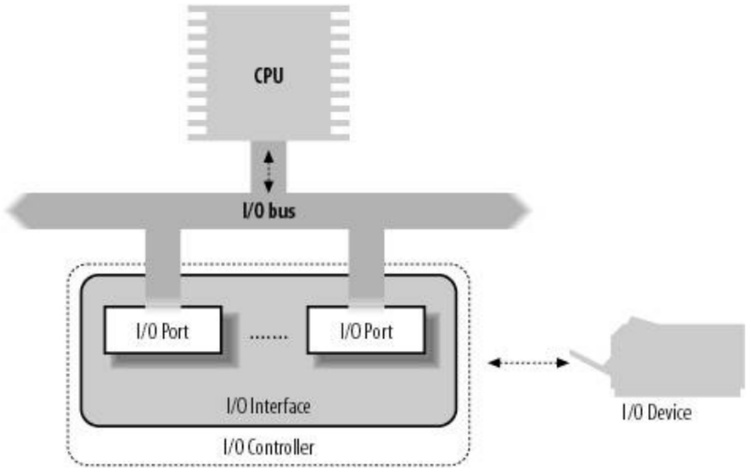
Peripheral Component Interconnect (PCI) is a local computer bus for attaching hardware devices in a computer and is part of the PCI Local Bus standard. Discover the PCI bus layout and hierarchy to see what devices are attached to the system (hint: `man lspci`)



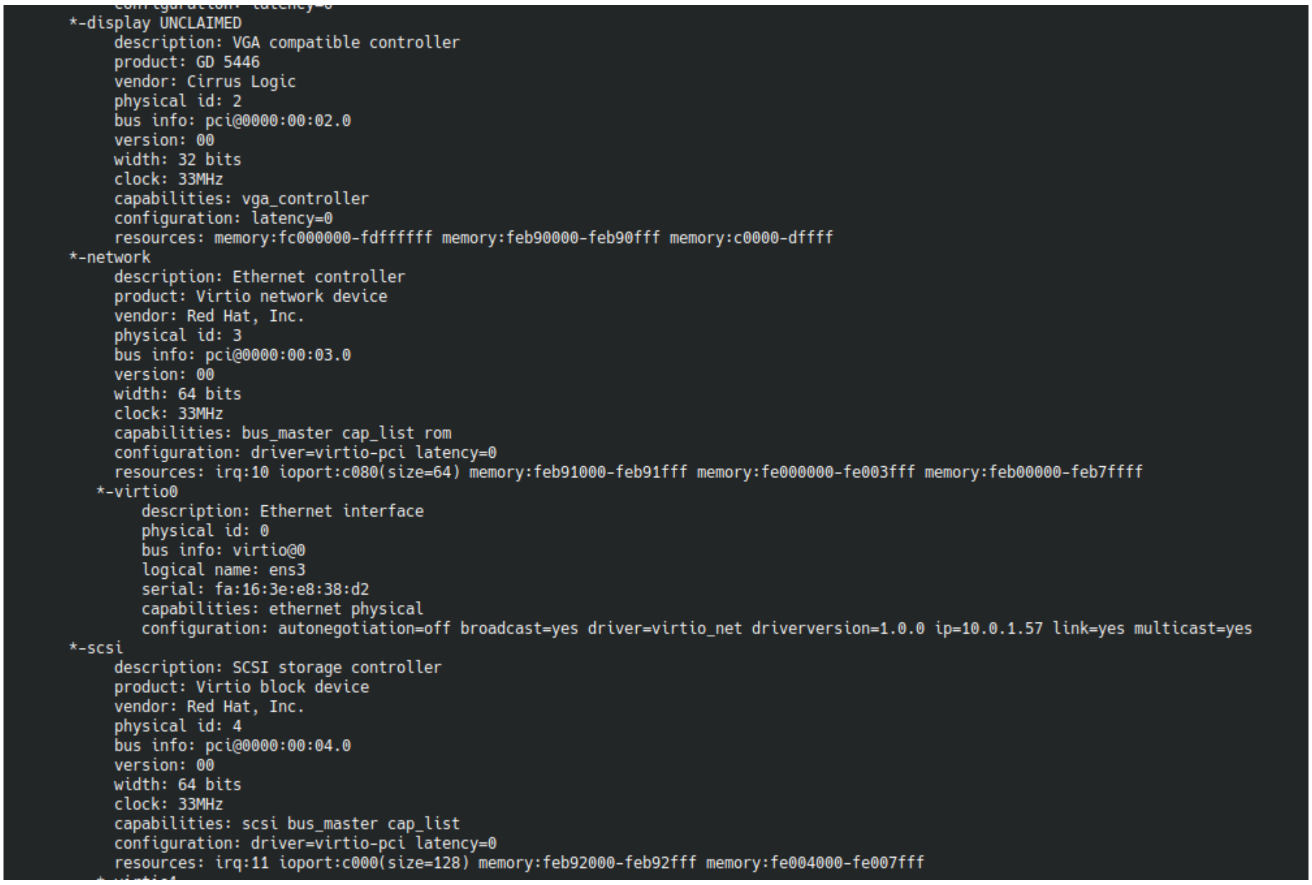
```
ubuntu@bsy-lecture:~/bsy/io$ lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
00:01.2 USB controller: Intel Corporation 82371SB PIIX3 USB [Natoma/Triton II] (rev 01)
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
00:02.0 VGA compatible controller: Cirrus Logic GD 5446
00:03.0 Ethernet controller: Red Hat, Inc. Virtio network device
00:04.0 SCSI storage controller: Red Hat, Inc. Virtio block device
00:05.0 Unclassified device [00ff]: Red Hat, Inc. Virtio memory balloon
00:06.0 Unclassified device [00ff]: Red Hat, Inc. Virtio RNG
```

Verify your understanding of the relations between all components by displaying the topology (component tree).

This figure below was introduced in the lecture. Based on the tools and your analysis above map the components of the figure to the hardware on the lab system. Which devices use IO ports and which IO memory mapping approach is implemented. Record some devices and their precise addresses used.



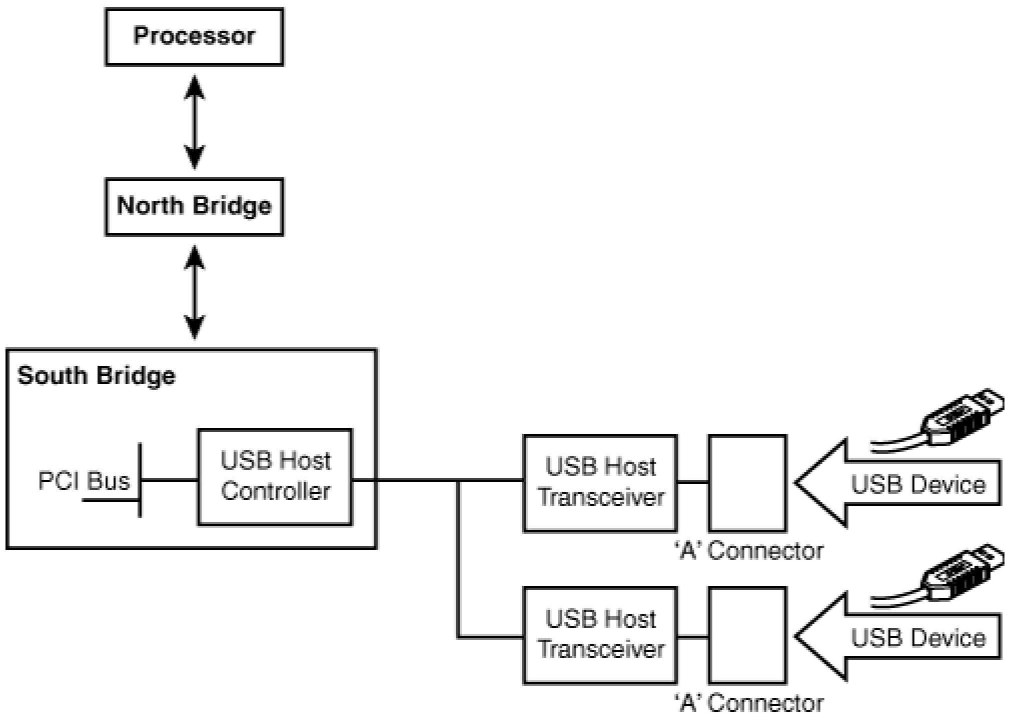
IO-Interface maps to Southbridge, IO Ports to devices connected to the SB may have IP ports or use Memory Mapped IO.



```
     *-display UNCLAIMED
          description: VGA compatible controller
          product: GD 5446
          vendor: Cirrus Logic
          physical id: 2
          bus info: pci@0000:00:02.0
          version: 00
          width: 32 bits
          clock: 33MHz
          capabilities: vga_controller
          configuration: latency=0
          resources: memory:fc000000-fdffffff memory:feb90000-feb90fff memory:c0000-dffff
     *-network
          description: Ethernet controller
          product: Virtio network device
          vendor: Red Hat, Inc.
          physical id: 3
          bus info: pci@0000:00:03.0
          version: 00
          width: 64 bits
          clock: 33MHz
          capabilities: bus_master cap_list rom
          configuration: driver=virtio-pci latency=0
          resources: irq:10 ioport:c080(size=64) memory:feb91000-feb91fff memory:fe000000-fe003fff memory:feb00000-feb7ffff
        *-virtio0
             description: Ethernet interface
             physical id: 0
             bus info: virtio@0
             logical name: ens3
             serial: fa:16:3e:e8:38:d2
             capabilities: ethernet physical
             configuration: autonegotiation=off broadcast=yes driver=virtio_net driverversion=1.0.0 ip=10.0.1.57 link=yes multicast=yes
     *-scsi
          description: SCSI storage controller
          product: Virtio block device
          vendor: Red Hat, Inc.
          physical id: 4
          bus info: pci@0000:00:04.0
          version: 00
          width: 64 bits
          clock: 33MHz
          capabilities: scsi bus_master cap_list
          configuration: driver=virtio-pci latency=0
          resources: irq:11 ioport:c000(size=128) memory:feb92000-feb92fff memory:fe004000-fe007fff
```

Find out what drivers are in use by the devices on the PCI bus. Which of these drivers are built-into the kernel and which ones are modules loaded on demand?

Universal Serial Bus has become the "universal" IO serial interface over the past decade. Analyze the configuration on your system and record hierarchy and features of USB devices that are attached and available to the OS (hint, start with: `man -k usb`).



Task 2 – Principles of IO

Physical vs Logical (Virtual) Devices
Double-check your hardware configuration, can you identify logical (virtual) IO devices? If, identify and analyse some and discuss their origin and features.

Synchronous vs Asynchronous Access
Understand the output of /proc/interrupts and discuss the columns in detail.

Print the evolution of this file in real-time onto your screen. Hint, you may find the "`watch`" tool useful.

Verify your understanding using "`lspci -vvv`", check for instance, the USB configuration. What can you tell about the interrupt (IRQ) configuration?

Shared IO Access
As soon as access to IO is shared, performance becomes a real issue. There are several tools to analyze IO performance, like iostat and iotop. Familiarize yourself with these two by running them and understanding their outputs.

Now create a scenario in which you create IO load. Try to understand and use the following command: "`wc /dev/zero &`" . Then analyze the impact via iostat and iotop for several scenarios. Record and discuss what you notice.

Now understand what the "`dd`" tool (man dd) is for. Now run a scenario in which you run several parallel instances of dd, each creating a file of 10G each. Before that make sure that you have enough disk space by creating a new volume via the Openstack user interface, attaching it to your Lab Instance, and mounting it into the filesystem, using /mnt as mountpoint. Hint, you can use "`dmesg`", "`fdisk`" to get info about your new virtual disk, and "`mkfs.ext4`" and "`mount`" to format and mount it.

On your new disk, create in parallel a number of files, using the following command. "`dd if=/dev/zero of=/mnt/iotest/testfile_n bs=1024 count=1000000 &`" What exactly is this command doing? What do you have to modify in order to use this command in parallel? Perhaps you want to run this as a shell script, simply put this command line-by-line into a text file and execute the text file (aka bash script file) on the shell. Once load is created, check IO performance. Discuss the results and verify your understanding.

Now experiment with the command “ionice”. Add different schedulers with different priorities to your operations and verify the result. Can you actually run one process faster than another? If yes, explain, if not explain too. Hint, “`less /sys/block/vXXX/queue/scheduler`” and `https://wiki.ubuntu.com/Kernel/Reference/IOSchedulers`

Blocking and Non-Blocking Devices

Interpret and understand the code below. What is the use case of the function select()?

```
#include <stdio.h>, #include <stdlib.h>, #include <sys/time.h>, #include
<sys/types.h>, #include <unistd.h>

int main(void)
{
    fd_set rfds;
    struct timeval tv;
    int retval;

   /* Set rfds to STDIN (fd 0) */
    FD_ZERO(&rfds); FD_SET(0, &rfds);

   /* Wait up to five seconds. */
    tv.tv_sec = 5;
    tv.tv_usec = 0;

   retval = select(1, &rfds, NULL, NULL, &tv);

   if (retval == -1)
        perror("select()");
    else if (retval)
        printf("OK");
        /* FD_ISSET(0, &rfds) will be true. */
    else
        printf("Not OK within five seconds.\n");

   exit(EXIT_SUCCESS);
}
```

Task 3 – Linux Device Model and UDEV

Navigate to /sys and familiarize yourself with the structure. Look for, and understand the content of the “uevent” file for the block device (volume) created for the IO performance task.

Enter the /dev directory and look for the device file that represents the volume. What can you see from the file attributes?

UDEV uses /sys to create devices files in /dev upon the appearance (e.g. hotplugged) of a device. List the rule files of UDEV that define this on-demand behavior. What happens if you attach a mouse as IO input device?

Remove the volume (detach it via OpenStack User Interface) created for the IO performance testing before and look for changes in the /sys and /dev directory.

Task 4 – Example IO Device: Hard Drive

`iostat` (see `man iostat`) is a useful tool to provide low-level disk statistics.

- Pick one hard drive of your system and display only the device utilization report in a human readable format and using megabytes per second.
- What does the output tell you?
- How can you only show one specific device?
- How can you show further extended details of that specific device i.e. sub-devices (hint: /proc)?
- Investigate in the `/proc` file system where disk statistics are taken by `iostat` for display.
- What is the `await` field and why is it important?
- If `rqm/s` is > 0 what does this indicate? What does it hint about the workload?

Now, measure your disk write output with `dd (c.f. man dd):`

```
dd if=/dev/zero of=speedtest bs=10M count=100
rm speedtest
dd if=/dev/zero of=speedtest bs=10M count=100 conv=fdatasync
```

- How can you explain the difference in output?
- What is a `fsync()` operation?

Cleanup!

IMPORTANT: At the end of the lab session:

- **Delete** all OpenStack VMs, volumes, security group rules that are no longer needed. You may want to keep some data for exam preparations.

Additional Documentation

OpenStack Horizon documentation can be found on the following pages:

- User Guide: https://docs.openstack.org/horizon/latest/