Queen Mary
University of London

ECS781P: Cloud Computing
Lab Instructions for Week 2

Overview of Operating Systems, Virtual Machines

Dr. Arman Khouzani, Dr. Felix Cuadrado

Jan 16, 2018

This is a "warm-up" set of exercises, reviewing some concepts from Operating Systems and Virtual Machines.

1 Operating Systems: Quick Overview

Make sure you have logged in to a Linux machine. Open a `terminal` and answer the following questions. Recall that you can invoke the documentation of a software by issuing the `man` command (short for manual).

1. Find out what each of the following commands does (from their man page), and what their output is on your machine.

   `pwd, ls, lsb_release, hostname, uname, whoami`

2. We can pass parameters to Linux commands typically by using a dash "-". For instance, execute and explain the difference between the following two commands: `ls` and `ls -l`.

3. Explain how the output of `uname -m` can be used to find out whether you are running a 32-bit or a 64-bit system.

4. We can of course have multiple parameters passed to a command too. What does `ls -l -a -S` do? (Note that Linux is case-sensitive, so `-s` is different from `-S`). Typically, we can combine the parameters for a shorter command, e.g. `ls -laS` is equivalent to `ls -l -a -S` (try it).

5. There are very strong "text" processing commands provided by Linux. Find out what each of the following does:

   `cat, head, tail, less, more, sort, uniq, comm, join, grep`

Among these, `grep` is perhaps the most useful: it can either be used to search for a text in a file, or can be passed the output of another command ("piped") for filtering based on keywords. For instance, try the following commands and explain what they do:

```
grep vendor /proc/cpuinfo
cat /proc/cpuinfo | grep vendor
man grep | grep case
dmesg | grep -i cert
```

process-management Each process has the following elements:

- **Process ID** (`PID`) a unique identifying number
- **Memory** a dedicated section of the memory, containing all the program instructions (*codes*), *stack* for keeping track of the function calls, *heap* for dynamic allocation (for variables that may need extra memory on the fly), and *static* segment for initialized global variables of the process.
- **Files** (files associated with the process, along with a file descriptor determining the read/write/execute permissions on each).
- **Registers** When the Kernel (temporarily) interrupts a process to serve another process (time-sharing) to achieve multi-tasking, the current state of the registers needs to be saved in order to be restored from exactly when the process was interrupted once the kernel resumes the process.
- **Kernel State** information about the process kept by the kernel, e.g. the state of the process (running, waiting for I/O, ready, sleeping (in the background), or terminated), priority level of the process, its performance and resource usage pattern (to use in its task scheduling for an improved overall performance).

1. The main command to collect information about the running processes is `ps`. Check its `man`, and explain what the following commands do:

   ```
   ps -ef
   ps -e r -u $(whoami)
   ps -o pid,uname,pcpu,pmem,comm -C firefox
   ps -e -o pid,args --forest
   ```

   (Compare the output of the last command to `pstree`).

2. Find the PID of the process that is run by you and is using the most portion of the memory and `kill` it, by sending a termination signal to it. Hint: a command like the following may help: `ps aux --sort=-%mem | head`

3. `top` is an "interactive" alternative to `ps` (If you are using your own laptop, install and run `htop` for a more pleasant-looking alternative). While top is running. hit (capital) `H` on your keyboard to toggle between task and thread-based statistics. How many tasks are being handled by the Kernel. How many threads? Which one is bigger? Why? How many of each is running/sleeping?

Access Control Linux keeps track of privileges (entailing limiting access, isolate processes), *etc.*through *control groups* or `cgroups` in short, which describes an association of specific permissions to each group and association of users to groups, and through `namespaces`, which abstract and isolate resources of different processes.

1. What does the first column of the output of `ls -l` show?
2. What does each of these commands do?

   `chmod, chown, chgrp`

3. First, create an empty text file by executing `touch hello.txt` (in a folder that you have the permission to write, e.g. in your home folder. If you are not sure where it is, run `echo $HOME`). Then see the output of `ls -l hello.txt`. What are the default permissions for the file that you just created?
4. Execute `chmod 666 hello.txt`, and then see the effect on the permissions.
5. Assume you want to give read and execute permission to your group (but not write), but only execute permission to everyone on your file. How should you modify the above command? Try it.
6. Try to remove all permissions from this file. Then try to delete the file (`rm hello.txt`). Why do you think you could still delete the file?

Accounting (monitoring and logging)

1. What does each of the following commands do?

   `who, w, id, users, last, uptime, lsof`

2. Use `ac` to find out the per day usage of the machine as well as per each individual user (using the appropriate parameter). Can you recognize all of these users?
3. Use `last` in combination with `ac` to find out how long the last immediate user has used your machine.

Memory/Storage management

1. Use `free -mt` (for units of megabytes) and `cat \proc\meminfo` to get the amount of free available memory.
2. Run `dmesg | grep -i memory` and interpret the result.
3. You can find out how much disk storage quota you are assigned and how much of it is left using the command `quota -s`. It is important to regularly check the output of this, to not run of out space. How much storage quota do you have right now and how much of it is left?
4. To find out which files/folders are taking the biggest size (especially if you have a limited disk quota) issue the following command:

```
du -h --max-depth=1 | sort -hr
```

What is the output? Does it matter which working directory you issue this command from? What does `--max-depth=1` signify? What does it change if you change this to `0`, or `2`?

File-System

1. **File-System** is a term describing how the OS names files and organizes them (places them logically) for storage and retrieval (to access them whenever needed). Open a terminal and change your current directory to root by typing `cd /`. Now write down the name of at least five *directories*, what they stand for, and what Unix/Linux uses them for. (Note: `usr` stands for Unix System Resources, not user!) You can get help from the *Linux Documentation Project* website: `http://www.tldp.org/index.html`. Specifically, what do we mean by the `binaries (bin)` and `libraries (lib)`?

2. Interpret the output of `lsblk`

3. Run `df -h` to get the statistics of the amount of used and available disk space per each file system type.

4. We can specifically find out where each command in Linux is (logically) stored by issuing the `which` followed by the name of the command. Find out where the file for the following commands are located: `ls, cp, rm, ssh, top, scp`. Did you get a feeling about where a command is likely to be found. For instance, before checking, make a guess where `rsync` and `gcc` are likely to be found. Were you correct in your guess?

5. See the content of `/usr/local/` what is being located here?

6. Check if the following commands are available on your machine, and if so, where are they stored (by issuing `which`):
`firefox, latex, python, matlab, mathematica, qemu-img`

I/O (Input/Output)

1. Interpret the output of the following commands: `lspci`, `lsusb`

Networking

1. What is the output of this command? `ip addr show`

2. Interpret the output of `ifconfig`

3. Interpret the output of `ethtool eth0`. Note that the `eth0` is the default name of the first Ethernet interface in Linux However, that may be different in your machine. You have found that out from interpreting the output of the previous command!

Basic Shell Scripting

1. We will be testing a sample `shell script`. Create a text file with your chosen name (e.g. `my_first_script.sh`) (each command is in a different line):

   ```
   cd ~
   mkdir ECS781P
   cd ECS781P
   touch my_first_script.sh
   ```

2. In the above commands, `touch` just creates an empty file. You can create and edit text files using one of the following programs: `vim, nano, emacs`, but if you prefer something very user-friendly, try `gedit` or `kwrite` (or something fancier like `subl` (sublime) or `atom`, or any of your favourite, but remember, not a word processor! (why?)).

3. Inside the text file just write the following, save and exit:

   ```
   #!/bin/bash

   echo "Hello World"
   ```

4. Give read and execute permission to all to your file (recall `chmod` was the command to use, with proper parameters).

5. execute the shell script simply by typing: `./my_first_script.sh`.

6. Now, for your second shell script, edit the text file to the following content and save it. **Note:** If you are copy/pasting the commands, beware that some characters like the single-quote ' may not paste properly. You need to fix it manually (it is with the @ key on UK keyboards!)

   ```
   #!/bin/sh

   free -m | awk 'NR==2{printf "Memory Usage: %s/%sMB (%.2f%%)\n", $3,$2,$3*100/$2 }'
   df -h | awk '$NF=="/"{printf "Disk Usage: %d/%dGB (%s)\n", $3,$2,$5}'
   top -bn1 | grep load | awk '{printf "CPU Load: %.2f\n", $(NF-2)}'
   ```

   Can you guess what it will do? Try it.

7. Edit your second shell script to show the amount of free memory in units of GB instead of MB. Try it and make sure the output is shown properly.

2 Virtual Machines: Intro

Virtualisation in general means, having a software that runs on a machine and *mimics* (*emulates*) another system with the same interface, such that any program/user that interacts with it would see the same outward behaviour irrespective of what the underlying machine is.
There are many types of virtualisation, depending on what is being "emulated" (e.g. Server Virtualisation, Desktop Virtualisation, Network Virtualisation, Data Virtualisation, Storage Virtualisation, Application Virtualisation, etc.).

When the emulated entity is an entire computer system, we call it a *virtual machine*. The software in charge of emulating computer systems, i.e., creating and running virtual machines is called a **virtual machine monitor (VMM)** or , as is more commonly known, a **hypervisor**. More on this will be discussed during lectures.
We are going to launch instances of virtual machines inside our "host" machine, and see the effect of it on its physical resources. We will be using a script named `vminstance`.

1. Find where the script of `vminstance` is located. *Hint:* use `locate` or `which` commands. Open the content of the file (using `cat` or `gedit` or `vim`, *etc.*). What language is it written in?

2. **Optional:** Try to understand as much of the script as you can. For instance, what does these lines do:

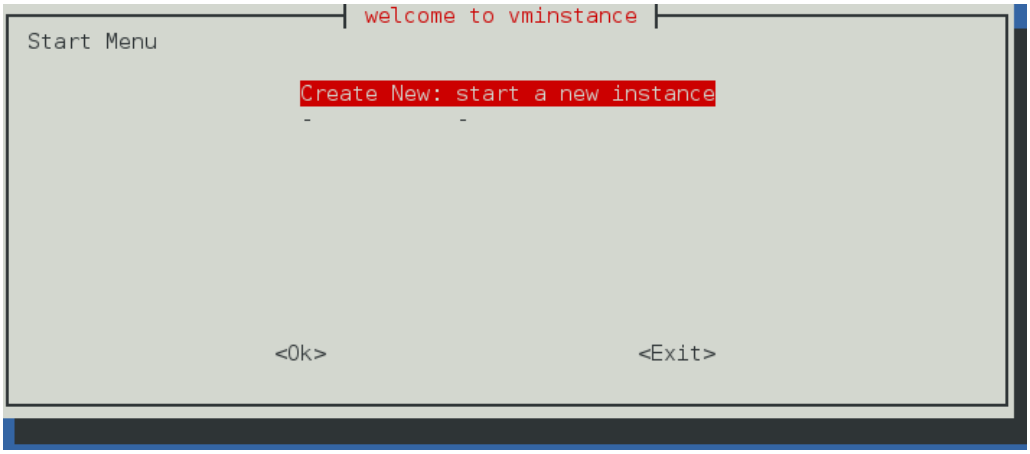
```
image_local="$HOME/Images"
image_base="/import/teaching/vminstance-images/"

###################
# Prereqs
###################
# Create Images directory
if [ ! -d $image_local ]; then mkdir $image_local ||\
 (echo "Error: Failed to create user Image directory" && exit 1); fi
# Check for base images
if [ ! -d $image_base ]; then \
 echo "Error: I cant see the source images at $image_base" && exit 1; fi
```

3. What does the folder of `/import/teaching/vminstance-images/` contain? What is a virtual machine's "*image*"?

4. Based on the script, what is the hypervisor that is being used? Hint: The most popular open-source hypervisors are Xen, VMware, OpenVZ, KVM, LXC and VirtualBox. And among the commercial ones: XenServer, Hyper-V, VMware vSphere, RHEV (Red Hat Enterprise Virtualisation), *etc.*.

5. There are two generic classes of hypervisors: *Type-I* and *Type-II*. Find out the difference between the two and answer: which type are we using right now?

6. Open two terminals. In one of them run `top`. In the other one run `vminstance` as we describe next.
   a) Run `vminstance` (type `vminstance` and then enter!)
   b) Delete any images that there are and you might have had from before (unless you are using them for other modules of course!). Now go to *Create New: start a new instance*. Choose your favourite OS from what is available. We suggest CentOS or Ubuntu for now.
   c) Run (connect to) the image.
   d) If you are prompted for a user-password, enter `localuser` for both (lower case, no space). Note that the password prompt for Linux does not show anything! (not even asterisks!)



e) **Caution:** You should not exit the VM manager before the VM is shut down (e.g. by running `sudo shutdown -h now` or `reboot -p` or `halt` or `poweroff` from a terminal in the VM, or the usual graphical way). Otherwise, you will lose all the changes you have made to the VM (including all the new software that you install and all your files that you develop).

7. All the new software that we install and files that we create will be stored if we shutdown properly. More precisely, for space efficiency, a file containing the changes (the difference) from the original image is saved locally, and next time the image can be relaunch from the previous state.

8. Run `cat /proc/cpuinfo` from inside the VM (virtual machine). What do you see (especially, for the *model name* entry).

9. When a VM is launched, what new processes in the host machine are running?

10. How much CPU, memory, and the free disk space does the VM have? How does it compare with the allocated resources to the hypervisor?

Communicating with a VM

1. Read the manual of `scp` and `rsync`. Use one of them to copy your script file that you created from the host machine to the VM machine, and from the VM machine back to the host machine.

   Note: you may need the ip address of the host and the VM. By now, you should know how to get them!

   Note if you get a connection refused prompt, it is because the ITL machines are protected behind a firewall. So you need to `sc` or `rsync` to a machine that you can log in from an outside computer (recall that our VM is now a completely independent machine). Examples:

   `bert.student.eecs.qmul.ac.uk`
   `grover.student.eecs.qmul.ac.uk`

   So to copy a `tst.txt` file (on the home directory) from your VM to your home directory on `grover.student.eecs.qmul.ac.uk`, you will enter the following in a terminal opened from your VM:

```
rsync ~/tst.txt your_EECS_usrname@grover.student.eecs.qmul.ac.uk:~
```

2. What "protocol" is being used for this transfer?

Scaling on a finite machine!

1. We will see what the effect of adding new VMs to a single machine with finite resources is! For this, start launching new VM instances while keeping track of the the CPU, memory, and the free disk space of our actual machine (host) after each instance of a virtual machine (guest OS) is launched. What is the effect with respect to the number of instances? **Caution: Too many VM instances will make the host machine unstable, especially, you may run out of memory or disk quota quickly. If that happens, your host OS will hang up. After all, you are not on a cloud yet!**

2. Note that you have root privilege on the VMs. So we are going to install a software called `stress` on them, which will put a controllable load on the resources (CPU, memory, I/O). You can install `stress` on Debian/Ubuntu by

   ```
   sudo apt-get install stress
   ```

   and on CentOS:

   ```
   sudo yum install epel-release
   sudo yum install stress
   ```

   Once you install `stress` on each of the VMs, run the following command to put an average load on the resources of each of them for 2 minutes:

   ```
   stress -c 2 -i 1 -m 1 --vm-bytes 128M -t 120s
   ```

   What is the effect on the resources of the host machine?

Tricky questions:

1. When you log in to a desktop, how can you determine whether it is an actual machine or a virtual machine?

2. Can you run a virtual machine inside a virtual machine?

Very Important Question By now, you should observe that virtualisation cannot create resources out of nothing! In fact, virtualisation definitely introduces some overhead in terms of usage of resources. This is natural: some of the resources are part of the VM itself, and some extra resources (CPU, Memory, storage, etc.) of the host is used for emulation, i.e., to manage and run the VMs itself. So the main question is, what is the advantage of virtualisation after all?