ECS781P: Cloud Computing
Lab Instructions for Week 2

Overview of Operating Systems, Virtual Machines

Dr. Arman Khouzani

Jan 20, 2017

1 Operating Systems: Intro

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

2 Virtual Machines: Intro

We are going to launch instances of OS (virtual machines) inside our machine, and see the effect of it on the physical resources. We will be using a script called `vminstance`.

- 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?

- 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
```

- What does the folder of `/import/teaching/vminstance-images/` contain?
- 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 Virtualization), *etc.*.
- Open two terminals. In one of them run `top`. In the other one run `\vminstance`, create and launch an instance of one of the OS images available. If prompted for username and password, try `localuser` for both.
- Run `cat /proc/cpuinfo` from inside the VM (virtual machine). What do you see (especially for the `model name` entry).
- When a VM is launched, what new processes in the host machine are running?
- How much cpu, memory, and the free disk space does the VM have? How does it compare with the allocated resources to the hypervisor?
- Keep track of the cpu, memory, and the free disk space of our actual machine (host) after each instance of a virtual machine (guest OS) is launched. How is the effect with respect to the number of instances?
- **Optional:** Note that you have root privillege 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 update` and then `sudo apt-get install stress`, and on CentOS: `yum check-update` and then `sudo yum install epel-release` and `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?