

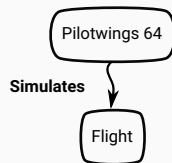
INTRODUCTION TO OS-LEVEL VIRTUALIZATION ON LINUX

Pedro Bruel

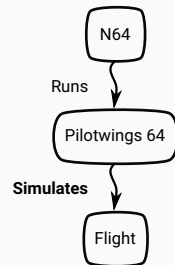
phrb@ime.usp.br

May 25th, 2020

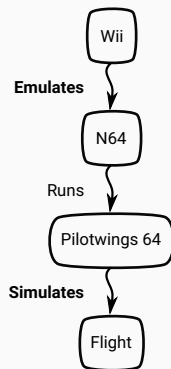
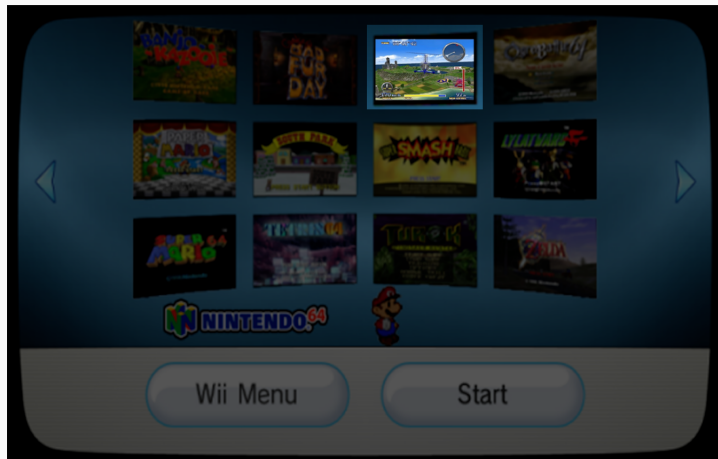
WHAT ARE SIMULATION, EMULATION, VIRTUALIZATION?



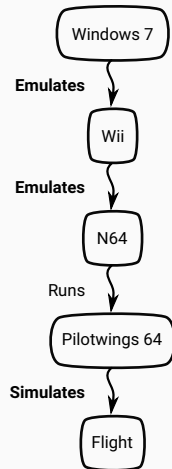
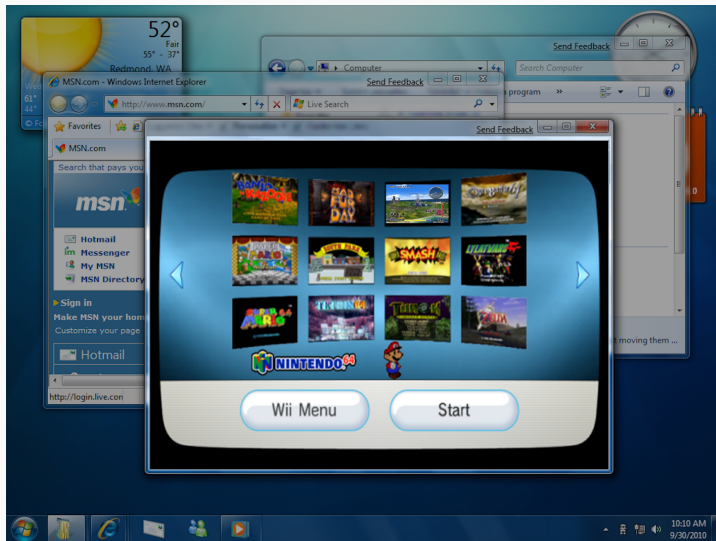
WHAT ARE SIMULATION, EMULATION, VIRTUALIZATION?



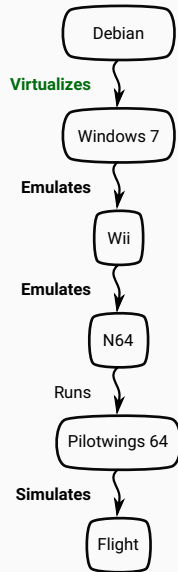
WHAT ARE SIMULATION, EMULATION, VIRTUALIZATION?



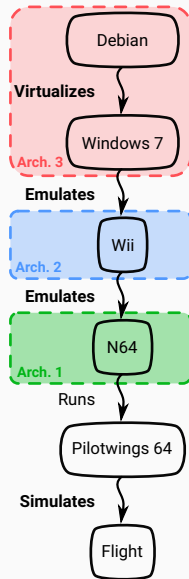
WHAT ARE SIMULATION, EMULATION, VIRTUALIZATION?



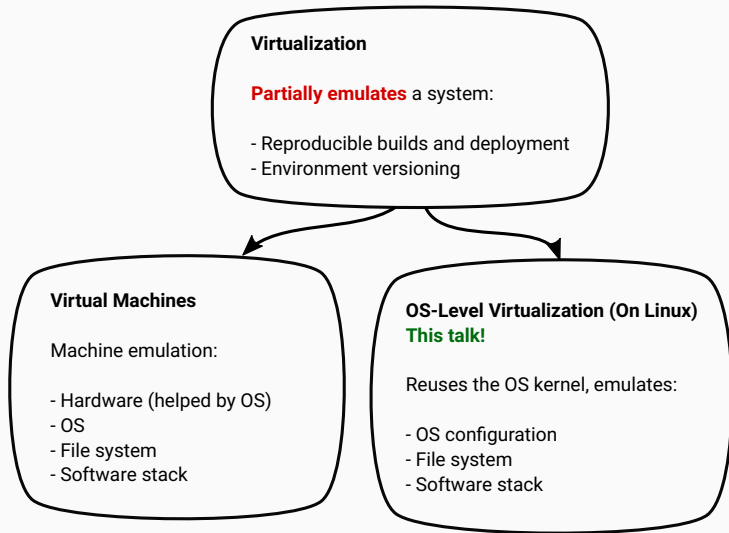
WHAT ARE SIMULATION, EMULATION, VIRTUALIZATION?



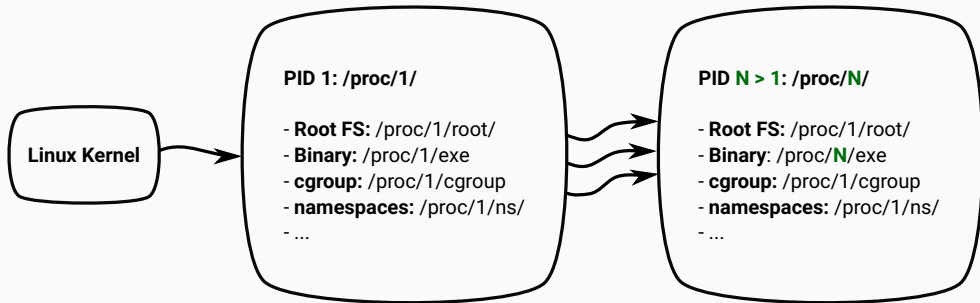
WHAT ARE SIMULATION, EMULATION, VIRTUALIZATION?



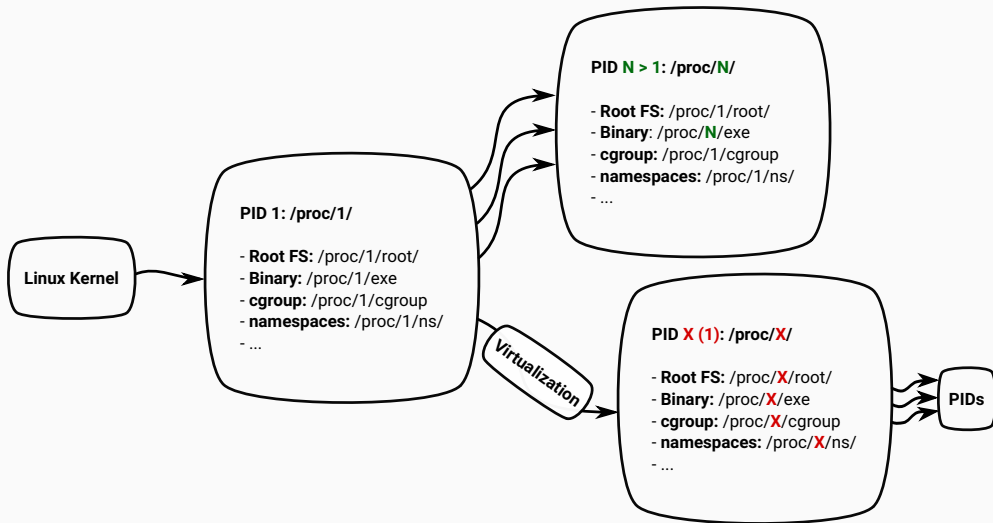
OS-LEVEL VIRTUALIZATION



OS-LEVEL VIRTUALIZATION ON LINUX



OS-LEVEL VIRTUALIZATION ON LINUX



HOW CONTAINERS WORK ZINE

Images used **with permission:**



Pedro Bruel @pedrobruel · 2h

Hey @b0rk, could I use pages 7 and 8 from your containers zine on an undergrad class on OS-level virtualization I'm making? Also, your zines are great!



 **Julia Evans** 
@b0rk

sure!

♡ 3 9:10 PM - May 14, 2020



containers = processes

7

a container is a group of Linux processes

on a Mac, all your containers are actually running in a Linux virtual machine

I started 'top' in a container. Here's what that looks like in ps:

outside the container				inside the container			
USER	PID	START	COMMAND	USER	PID	START	COMMAND
root	23540	20:55	top	root	25	20:55	top
bork	23546	20:57	top				

these two are the same process!

container processes can do anything a normal process can ...

I want my container to do X Y Z W!

sure! your computer, your rules!

... but usually they have restrictions

different PID namespace

different root directory

cgroup memory limit

limited capabilities

not allowed to run some system calls

the restrictions are enforced by the Linux kernel

NO, you can't have more memory!

on the next page we'll list all the kernel features that make this work!

container kernel features

8

containers use these Linux kernel features

"container" doesn't have a clear definition, but Docker containers use all of these features.

♥ pivot_root ♥

set a process's root directory to a directory with the contents of the the container image

★ cgroups ★

limit memory/CPU usage for a group of processes



♥ namespaces ♥

allow processes to have their own:

- network
- PIDs
- hostname
- mounts
- users
- + more

★ capabilities ★

security: give specific permissions

♥ seccomp-bpf ♥

security: prevent dangerous system calls

★ overlay filesystems ★

this is what makes layers work! Sharing layers saves disk space & helps containers start faster

CONTAINERS FROM SCRATCH: OBTAINING AN "IMAGE"

```
#!/usr/bin/bash
```

```
IMG_DIR="alpine_img"
```

```
IMG_REPO="https://us.images.linuxcontainers.org/images"
```

```
IMG_URL="$IMG_REPO/alpine/3.11/amd64/default/20200521_13:00/rootfs.tar.xz"
```

```
[ ! -d $IMG_DIR ] && \
```

```
    mkdir -p $IMG_DIR && \
```

```
    curl $IMG_URL | tar xJ -C $IMG_DIR
```

```
cd $IMG_DIR
```

CONTAINERS FROM SCRATCH: CREATING CGROUPS AND SETTING LIMITS

```
CGROUP_ID="MAC0475-145"  
sudo cgcreate -g "cpu,cpuacct,memory:$CGROUP_ID"  
sudo cgset -r cpu.shares=512 "$CGROUP_ID" # 1024 is 100% CPU  
sudo cgset -r memory.limit_in_bytes=10000000000 "$CGROUP_ID"
```

CONTAINERS FROM SCRATCH: LAUNCHING AN ALPINE CONTAINER

```
HOSTNAME="alpine-container"
sudo cgexec -g "cpu,cpuacct,memory:$CGROUP_ID" \
  unshare -fmuipn --mount-proc \
  chroot "$PWD/" \
  /bin/sh -c "PATH=/bin && mount -t proc proc /proc && hostname $HOSTNAME && sh"
```