

Advanced Operating Systems

Lab Tools Intro

2017/18

www.ce.unican.es

vpuente@unican.es



Environment

- ▣ (Some) current tools as a “regular” user
 - ◆ Virtual Machine deployment/provisioning system (vagrant)
 - ◆ Version control system (git)

Vagrant

- It's a abstraction layer on top of any virtualization layer
 - ◆ Works with VirtualBox, Vmware, Hyper-V, ..., lxc, docker, etc...
 - ◆ Although the lab work can be done with plain VirtualBox, vagrant is recommended to avoid the hassle of crude virtual machines (VM)
 - ◆ Used as a "simplification" tool for VM handling

- Recipe
 1. Install vagrant and VirtualBox (Linux/Windows/OSX)
 2. `mkdir myWorkingDir`
 3. `vagrant init debian/jessie64`
 4. `vagrant up`
 5. `vagrant ssh !`

How to work with vagrant

- ▣ Boxes != instances
- ▣ List boxes
 - ◆ `vagrant box list`
- ▣ Create a new instance (persistent)
 - ◆ `mkdir directory; cd directory; Vagrant init some_box`
- ▣ File interaction
 - ◆ Use the shared dir and work in the host
 - `/vagrant` directory is working directory in host
 - Requires VirtualBox utils installed in the VM (version number should match)
 - Beware clock-skew between VM and host (might affect make/git)
 - ◆ Use x11 forwarding (required a X11 server)
 - ◆ Use rdp (requires Windows host and a rDesktop/VNC client)
 - ◆ Use VirtualBox interface

Other useful commands

▣ To handle instances

- ◆ To inspect the system status
 - `vagrant global-status`
- ◆ To delete instances use "`vagrant destroy`" (never delete a VM from virtualBox interface!)


▣ To handle boxes

- ◆ `vagrant box list` -- List versions installed, that might me updated in remote server
- ◆ `vagrant box update` -- To download updated versions of the box (v.gr. if the box is upgraded on atlas.hashicorp.com)
- ◆ `vagrant box remove vpuente/AOSUC1617 --box-version 1.1` -- cleans old version for that box

AOS Vagrant Box: vpuente/AOSUC1617 (Demo)

Vagrant box vpuente/AOSUC - Vagrant Cloud

Inbox - vpuente@gmail.com

 HashiCorp Vagrant Cloud

DashboardSearchPricing

VagrantHelpvpuente ▾

vpuente / AOSUC Vagrant box

VersionsNew VersionSettingsAccess

How to use this box with Vagrant:

VagrantfileNew

```
Vagrant.configure("2") do |config|
  config.vm.box = "vpuente/AOSUC"
end
```

v1.22

 currently released version

Edit

This version was created 12 months ago.

Adds support for support graphical debug:

- ddd
- X11 forwarding to.

To use X it u need a host running linux or macOS. In windows host you can install [mobaXterm](#).

1 provider for this version.

Add a provider

virtualbox

 Hosted by Vagrant Cloud (943 MB)

Edit

v1.21

Edit

This version was created about 1 year ago.

Includes git and configuration

1 provider for this version.

Add a provider

virtualbox

 Hosted by Vagrant Cloud (935 MB)

Edit

Advantages

- Allow to provision the Environment without tinkering with the files
 - ◆ Download the box "once"! (not each time you install it)
 - ◆ Use it many times in many different contexts (v.gr. a particular lab or section)
 - Share the changes (labs, exams, etc...)
- Allows to run the box to any provider (beyond VirtualBox)
 - ◆ Local or external (i.e. Cloud provider such as AWS, GCE, Azure, etc...)
 - ◆ A higher level of automation are Chef and Puppet (automated delivery and provisioning)
 - ◆ Plays nice with Docker, CI systems (github, gitlab), ...
- Only used as a tool!
 - ◆ Interesting in learning more?: Sistemas Virtualización y Seguridad

There are other options?

- ▣ It's possible to do the lab natively in:
 - ◆ Linux (ugh)
 - ◆ Windows 10
 - Using linux subsystem
 - https://msdn.microsoft.com/es-es/commandline/wsl/install_guide
 - ◆ osX
 - Using port or brew
 - <https://stackoverflow.com/questions/39052271/compile-xv6-on-mac>
- ▣ My advice?
 - ◆ Pick your choice...

▣ Code editor

- ◆ Vim ☺
- ◆ **Visual Source Code**
- ◆ SublimeText
- ◆ Atom
- ◆ ...

▣ Debugger

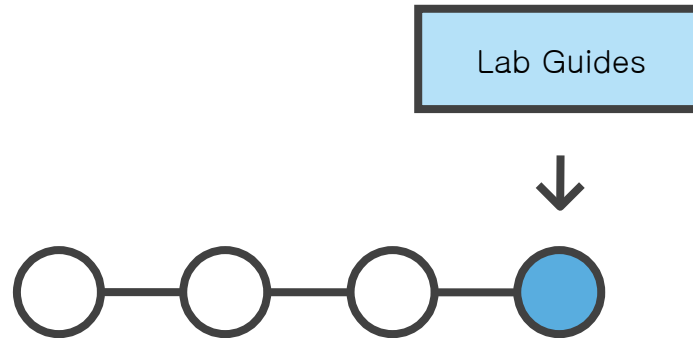
- ◆ Remote-debug (to QEMU)
- ◆ gdb <http://beej.us/guide/bggdb/>
- ◆ Other gdb frontends (ddd, VSC, etc...)

Lab work

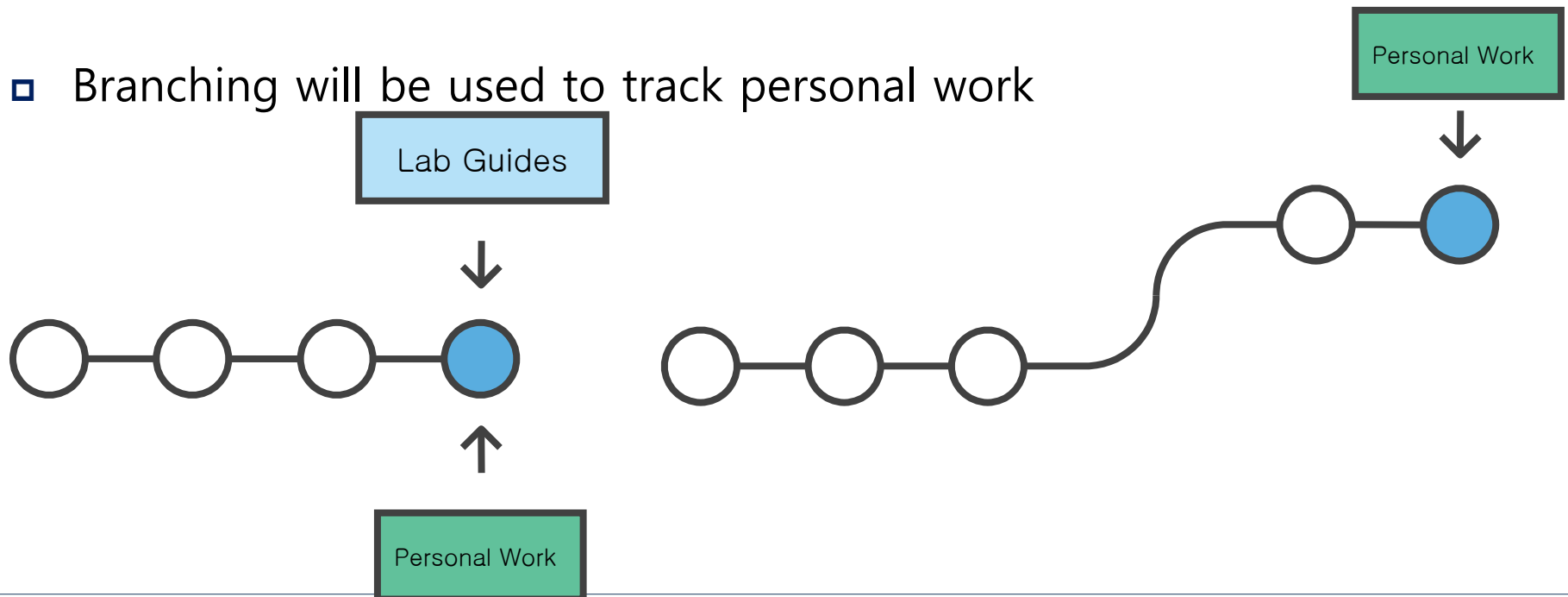
- ▣ All lab material will be available in git lab repository
 - ◆ git clone <https://gitlab.com/AOSUC/Lab.git>
- ▣ Material
 - ◆ Where you have to work
- ▣ Reference
 - ◆ OSTEP original material (ostep.org)
- ▣ We will work using a simple git branch+up-stream based workflow
 - ◆ <https://www.atlassian.com/git/tutorials/what-is-git>

Personal Work tracking

- Git repo contains a “chain” of atomic changes called commits



- Branching will be used to track personal work

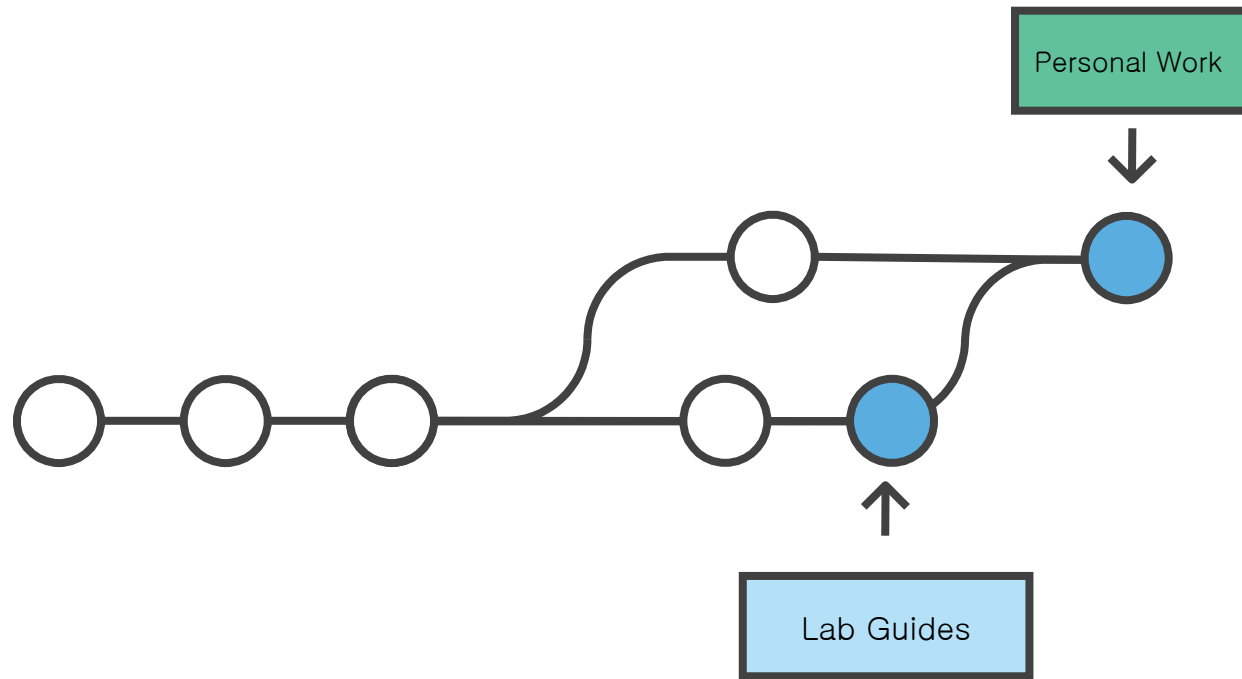


Workflow

- ❑ Create an account in gitlab.com
- ❑ Create a private repository (for lab)
- ❑ Fork <https://gitlab.com/AOSUC/Lab.git>
- ❑ Invite vpuente@gmail.com to such repository
- ❑ Start working
 - ❑ `git clone https://username@gitlab.com/Lab.git`
 - ❑ `git branch <DNI>`
 - ❑ `git checkout <DNI>`
 - ❑ START WORKING THERE / COMMIT AS MUCH AS U NEED!!!
 - ❑ `git add NEW_FILES` **DO NOT MODIFY ALREADY PRESENT FILES (merge conflicts)**
 - ❑ **<WORK>**
 - ❑ `git commit <WORK>` `git commit <WORK>` `git commit <WORK>` ...
 - ❑ `git push`
 - ❑ `git pull` <https://username@gitlab.com/AOSUC/Lab.git>

Updating of Guides (Pull)

- ▣ Merging updates in material

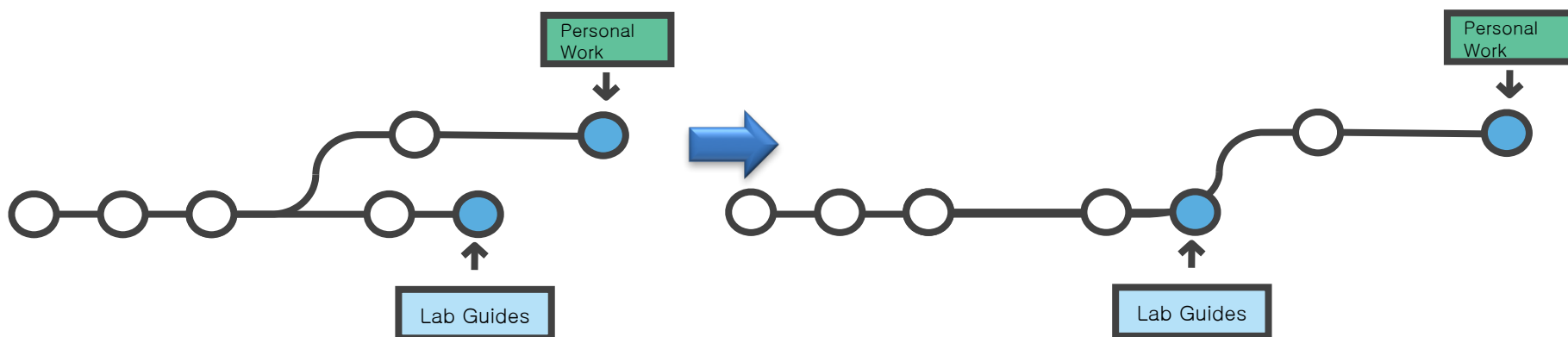


- ▣ Allow the professor to track your work
- ▣ Allow to automatize C&P detection

Download changes from the common repo

■ How to integrate changes from mainline in my fork?

- ◆ `git checkout master`
- ◆ `git pull https://gitlab.com/AOSUC/Lab`
- ◆ `git checkout <DNI>`
- ◆ `git rebase master`



■ This process might fail if done in “non” reliable clock environment

- ◆ For example if we have clock-skew between system and files (v.gr. Virtual Box and host shared directory !!!)

Next level (not required but advised to try at least)

- ▣ Create additional branches for a particular work
- ▣ Merge with DNI branch when done
 - ◆ Only use the Web interface to merge
 - ◆ Command line tool are not easy to use (especially when conflicts appears)
- ▣ Allows to work on multiple tasks at once (and do not have a chaos in hands)
- ▣ Use issue board to track your progress

Gitlab Interface

(not needed git deep understanding)

The screenshot displays the GitLab web interface. At the top, the 'AOSUC' group page is visible, featuring navigation tabs for Group, Activity, Milestones, Issues, Merge Requests, Members, and Contribution Analytics. Below this, the '@AOSUC' profile is shown with the University of Cantabria logo and name. A sidebar on the left lists projects: Lab, Biblio, Management, and Material. The main content area shows the 'Lab' project repository. It includes a table of files (HW, .gitignore, README.md, get_all.sh) and a 'Lab and Homework' section with links to HW OSTEP material and Guides. A terminal window is overlaid on the bottom right, showing the output of git commands.

```
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 403 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@gitlab.com:AOSUC/Lab.git
cc9a30f..b33b3e1 master -> master
cigal LAB (master)$ ls
HW README.md get_all.sh
cigal LAB (master)$ git log
commit b33b3e17b6340531b21e3a31c6324856c0c64061
Author: Valentin Puente <vpuente@unican.es>
Date: Fri Aug 26 16:40:48 2016 +0200

Readme

commit cc9a30f7ef89d32674d49514aa53ed51bce2be7a
Author: Valentin Puente <vpuente@unican.es>
Date: Fri Aug 26 16:38:25 2016 +0200

Initial commit
cigal LAB (master)$
```


GitLab interface (Demo)

The screenshot displays the GitLab web interface. The top navigation bar includes links for Project, Activity, Repository (selected), Pipelines, Registry, Graphs, Issues (0), Merge Requests (0), and Wiki. Below this, a secondary navigation bar shows Files (selected), Commits, Network, Compare, Branches, Tags, and Locked Files. The main content area is divided into two panels. The left panel shows a file tree with folders P0 through P5, each containing an 'Ops' file. The right panel shows the content of the 'README.md' file, which includes a title 'Project 0: Intro to the lab & Warmup', a table of contents, objectives, an overview, and a list of steps.

Project Activity **Repository** Pipelines Registry Graphs Issues 0 Merge Requests 0 Wiki

Files Commits Network Compare Branches Tags Locked Files

master Lab / Projects / +

Q Find File

Name	Last Commit > 6e981c07 - 11 days ago Fix tables	History	Lock	Last Update
..				
P0	Ops			11 days ago
P1	Expand P0 with C example			
P2	Expand P0 with C example			
P3	Expand P0 with C example			
P4	Expand P0 with C example			
P5	Expand P0 with C example			

master Lab / Projects / P0 / +

Q Find File

Name	Last Commit > 6e981c07 - 11 days ago Fix tables	History	Lock	Last Update
..				
README.md	Ops			11 days ago

README.md

Project 0: Intro to the lab & Warmup

[TOC]

Objetives

- Prepare the information structure in gitlab.com
- Familiarize yourself with the working environment
- Refresh your C knowledge and tools use

Overview

LAB designed to get used to the tools used in forthcoming labs and refresh OS course tools.

Steps

1. Accept the invitation to gitlab. Create and account there.
2. Fork the material repository in a private repository, creating a branch with your DNI.
3. Invite to such repository to vpunte@gmail.com
4. Wait for all
5. Update your repository with the new update
6. Rebase your code
7. Create a vacant instance in the top level directory

Structure of the repository (Demo)

□ Homework's

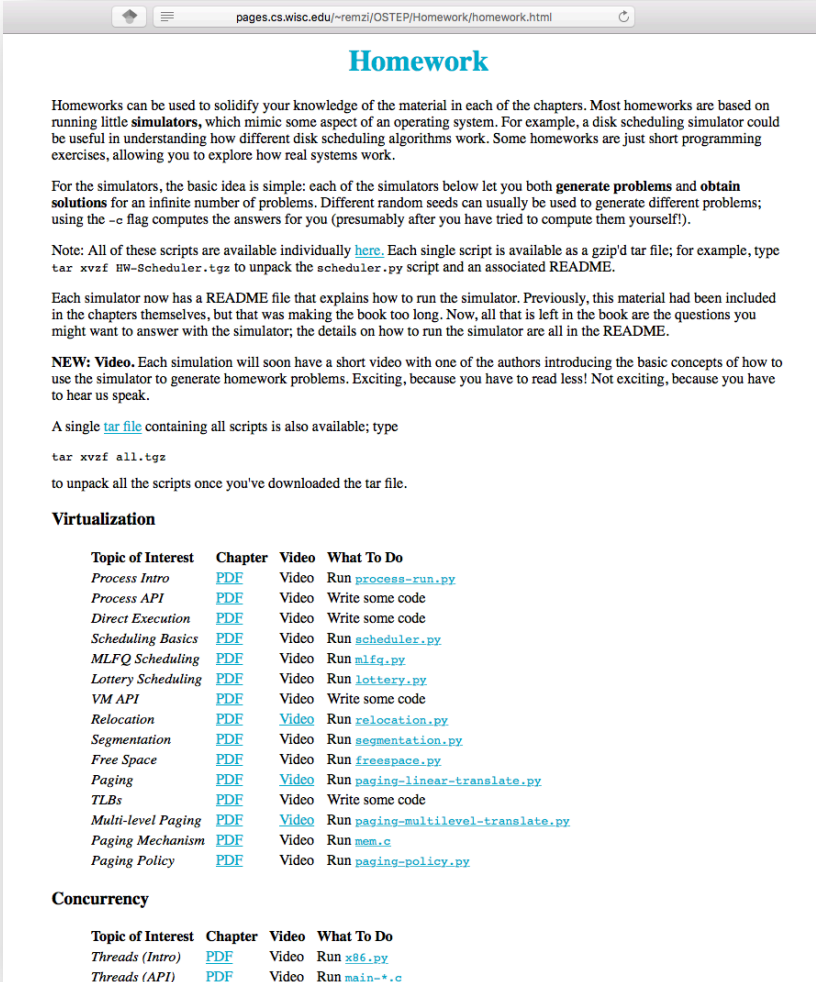
- ◆ Book material

□ Projects

- ◆ Develop here. Add a SOLUTION.md at the end on the P{\$\$\$} dir.

□ xv6

- ◆ Source code if the hacking environment used



The screenshot shows a web browser window with the address bar displaying `pages.cs.wisc.edu/~remzi/OSTEP/Homework/homework.html`. The page title is "Homework" in blue. The content includes several paragraphs of text explaining the purpose of homeworks and how to use the simulators. It also provides instructions on how to unpack the scripts using `tar` and `xvzf`. A section titled "Virtualization" contains a table with columns: "Topic of Interest", "Chapter", "Video", and "What To Do". The table lists various topics such as "Process Intro", "Process API", "Direct Execution", "Scheduling Basics", "MLFQ Scheduling", "Lottery Scheduling", "VM API", "Relocation", "Segmentation", "Free Space", "Paging", "TLBs", "Multi-level Paging", "Paging Mechanism", and "Paging Policy". Each row provides links to PDFs, videos, and instructions on what to do. A "Concurrency" section follows with a similar table listing "Threads (Intro)" and "Threads (API)".

Homework

Homeworks can be used to solidify your knowledge of the material in each of the chapters. Most homeworks are based on running little **simulators**, which mimic some aspect of an operating system. For example, a disk scheduling simulator could be useful in understanding how different disk scheduling algorithms work. Some homeworks are just short programming exercises, allowing you to explore how real systems work.

For the simulators, the basic idea is simple: each of the simulators below let you both **generate problems** and **obtain solutions** for an infinite number of problems. Different random seeds can usually be used to generate different problems; using the `-c` flag computes the answers for you (presumably after you have tried to compute them yourself!).

Note: All of these scripts are available individually [here](#). Each single script is available as a gzip'd tar file; for example, type `tar xvzf HW-Scheduler.tgz` to unpack the `scheduler.py` script and an associated README.

Each simulator now has a README file that explains how to run the simulator. Previously, this material had been included in the chapters themselves, but that was making the book too long. Now, all that is left in the book are the questions you might want to answer with the simulator; the details on how to run the simulator are all in the README.

NEW: Video. Each simulation will soon have a short video with one of the authors introducing the basic concepts of how to use the simulator to generate homework problems. Exciting, because you have to read less! Not exciting, because you have to hear us speak.

A single [tar file](#) containing all scripts is also available; type

```
tar xvzf all.tgz
```

to unpack all the scripts once you've downloaded the tar file.

Virtualization

Topic of Interest	Chapter	Video	What To Do
<i>Process Intro</i>	PDF	Video	Run process-run.py
<i>Process API</i>	PDF	Video	Write some code
<i>Direct Execution</i>	PDF	Video	Write some code
<i>Scheduling Basics</i>	PDF	Video	Run scheduler.py
<i>MLFQ Scheduling</i>	PDF	Video	Run mlfq.py
<i>Lottery Scheduling</i>	PDF	Video	Run lottery.py
<i>VM API</i>	PDF	Video	Write some code
<i>Relocation</i>	PDF	Video	Run relocation.py
<i>Segmentation</i>	PDF	Video	Run segmentation.py
<i>Free Space</i>	PDF	Video	Run freespace.py
<i>Paging</i>	PDF	Video	Run paging-linear-translate.py
<i>TLBs</i>	PDF	Video	Write some code
<i>Multi-level Paging</i>	PDF	Video	Run paging-multilevel-translate.py
<i>Paging Mechanism</i>	PDF	Video	Run mem.c
<i>Paging Policy</i>	PDF	Video	Run paging-policy.py

Concurrency

Topic of Interest	Chapter	Video	What To Do
<i>Threads (Intro)</i>	PDF	Video	Run x86.py
<i>Threads (API)</i>	PDF	Video	Run main-*.c