Xv6 Presentation

Presented By: Blake Collins, Mark Neitzel, & Taylor Perry

What Are We Doing???

- We are doing Xv6
- Sixth Edition Unix a.k.a. Version 6 Unix
- First public release of Unix out of Bell Labs
- Designed for DEC PDP-11*
- Original source code still available:

http://minnie.tuhs.org/cgi-bin/utree.pl

- The entire source code can be converted to a PDF via "make print"

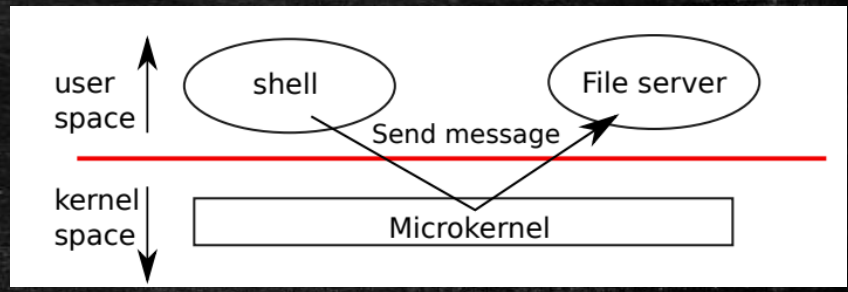
Info taken from: https://www.slideshare.net/ZubairNabi/aos-lab-2-hello-xv6

Why Is It Needed???

- 1). It's not, but…
- 2) It was created by MIT's OS Engineers to hopefully replace archaic Kernel environments.
- 3) Very easy to install. 4 EASY STEPS *more if you don't already have a VM set up
- 4) Do you not like options???
  -you have MINIX, Nuttx, UNICOS, etc..

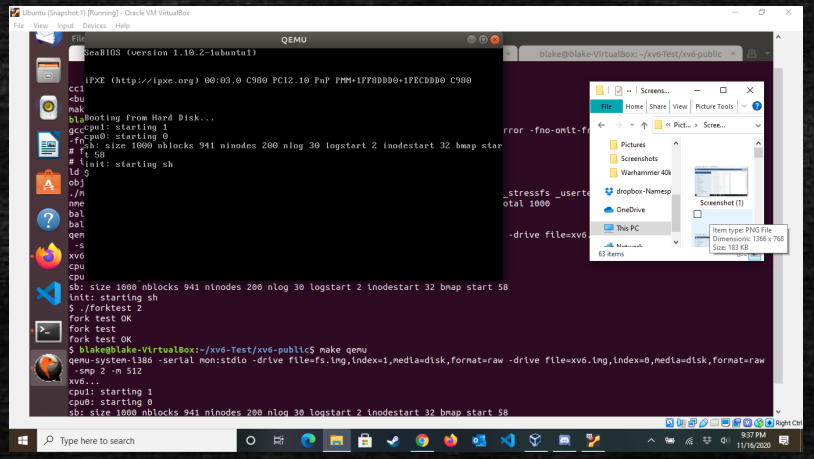
Xv6 Architecture

- Xv6 is no longer being serviced and is instead succeeded by RISC-V
  - Runs on a multi-core RISC-V microprocessor
- For each program you want to run it needs its own specific library.
- Uses 38 bits when running virtual memory. Making the max address 2^38 –1
- It is possible to add header files but requires a lot of coding
- Switches from User space to kernel space.
  - –Operations are done quickily but at the cost of security.
    - Easy to mess up the kernel if a user
- Xv6 is written in LP64 meaning long and pointers are in 64 bit, but int is in 32 bit
- Expects multicore hardware where multiple CPU's share memory but execute programs in parallel.

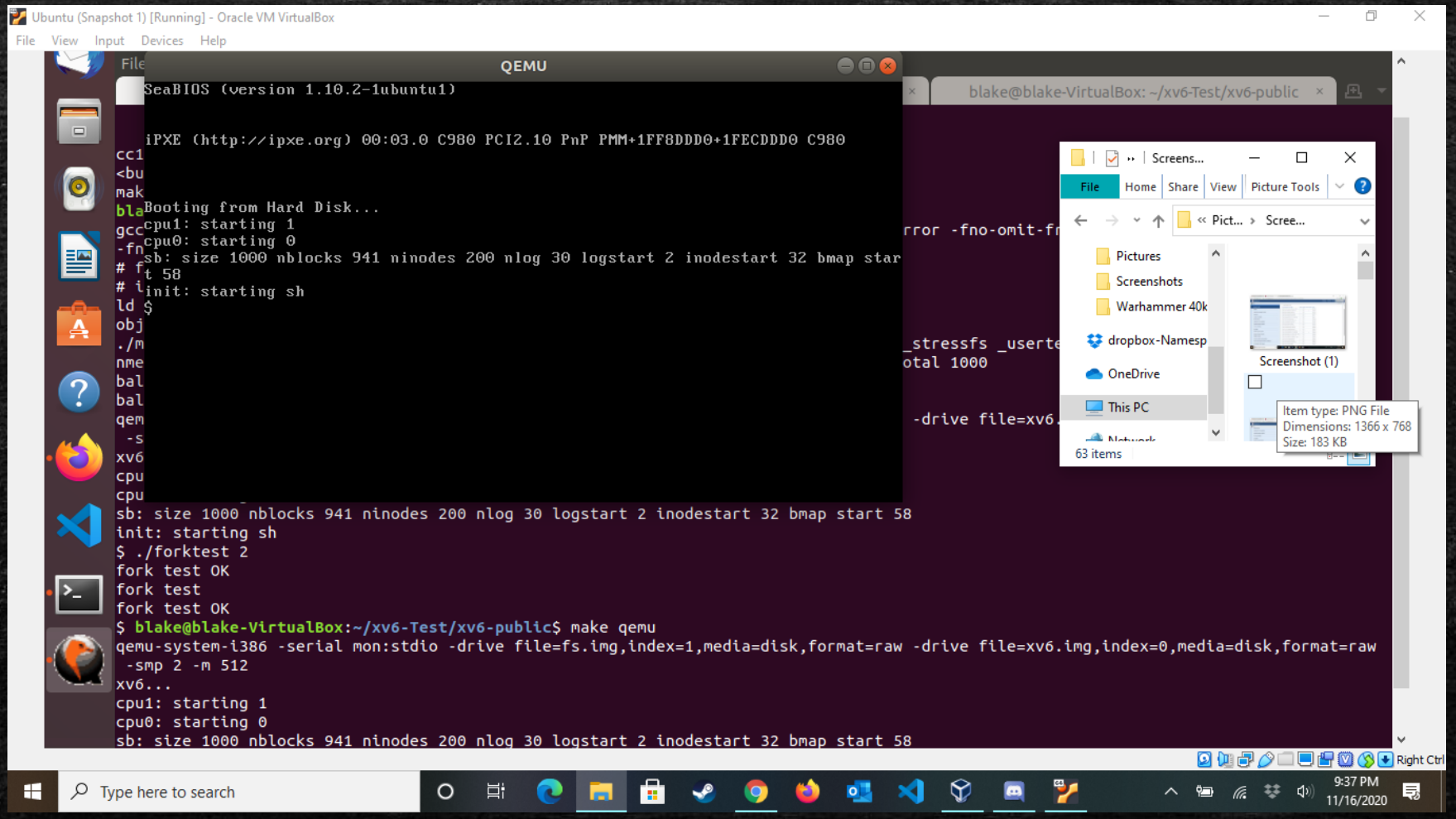


How Do You Implement???

- Initial implementation is rather easy, all you need to do is have qemu installed, and then clone the vx6 repository.
- git clone git://github.com/mit-pdos/xv6-public.git xv6
- To make an initial launch, first input "make" and then "make qemu" and it will pull up a terminal with qemu on it and the programs that have been made for it.



How Do You Implement??? Cont...

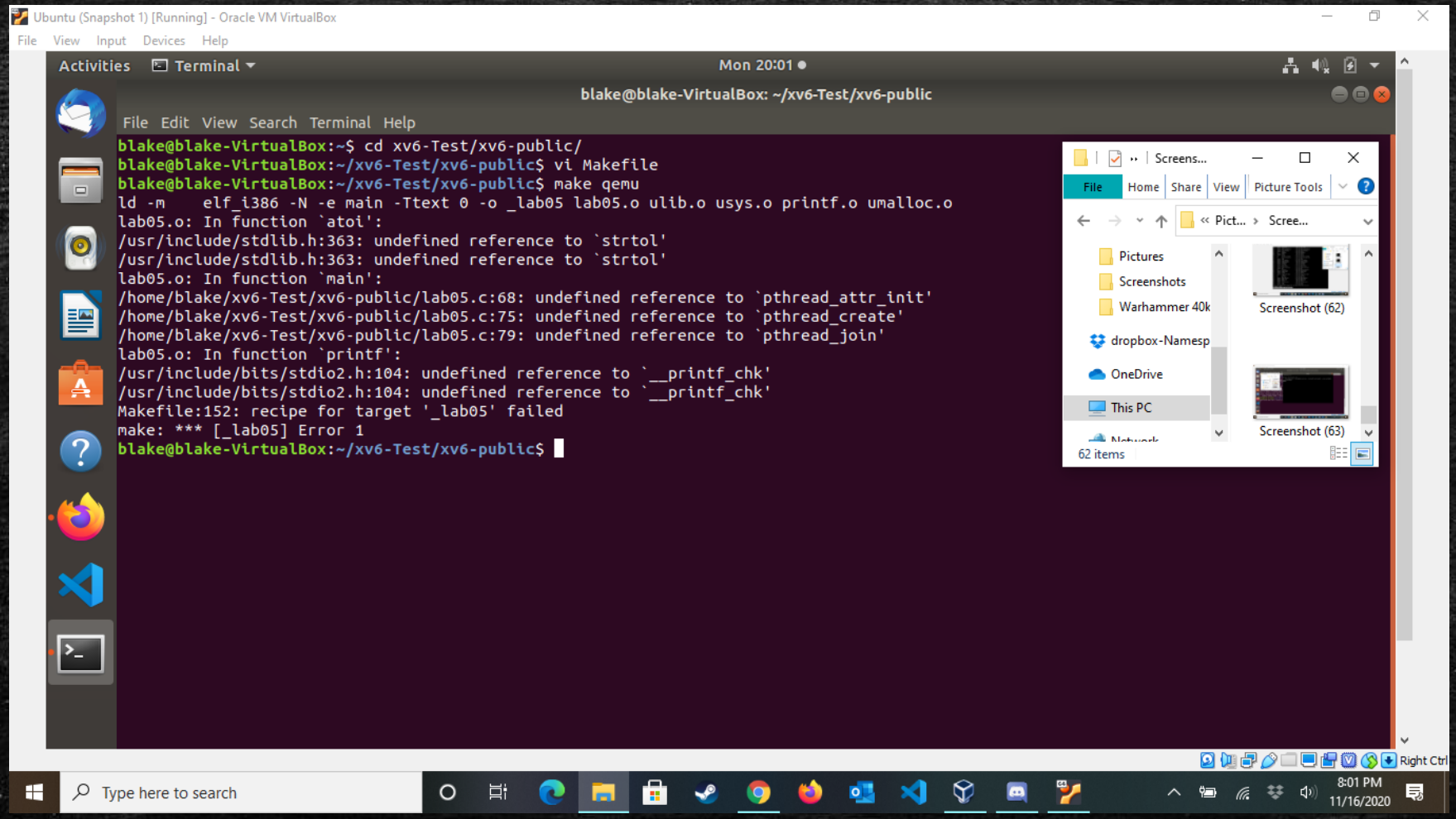


Issues & Problems Occurred

- Many steps to implement anything. For a system call you have to

1. Go into the syscall.h and add your new call name and its number
2. Go to sysproc.c file to add what you want to do
3. Then go to syscall.c to add your syscall to the syscall table
4. Now you can create a file that creates what you want your syscall to do. (Print hello, sleep, mkdir, etc..)
5. Furthermore add your syscall to the user.h header file so it can be seen
6. Write a .c file that wraps up your syscall. This contains all the code
7. FINALLY you can compile your xv6 and your syscall will initiate.

Issues & Problems Occurred Cont...



Should This Be Included In Future OS Classes???

- IT DEPENDS!!!
- If there is enough time to learn how to use it, then it would be beneficial to be included in future OS classes.
- If there isn't though, then this is very rough to handle with minimal instruction.

No work, All Play???