# OSN Tutorial 5

Introduction to XV6

## What is XV6?

- A “toy” operating system developed at MIT
- Based on Unix v6 (hence the name)
- Provides the basic interfaces introduced in Unix and also mimics Unix’s internal design.
- Processes, files and directories, paging, RR scheduling, interrupts...

# Installation

https://pdos.csail.mit.edu/6.S081/2020/tools.html

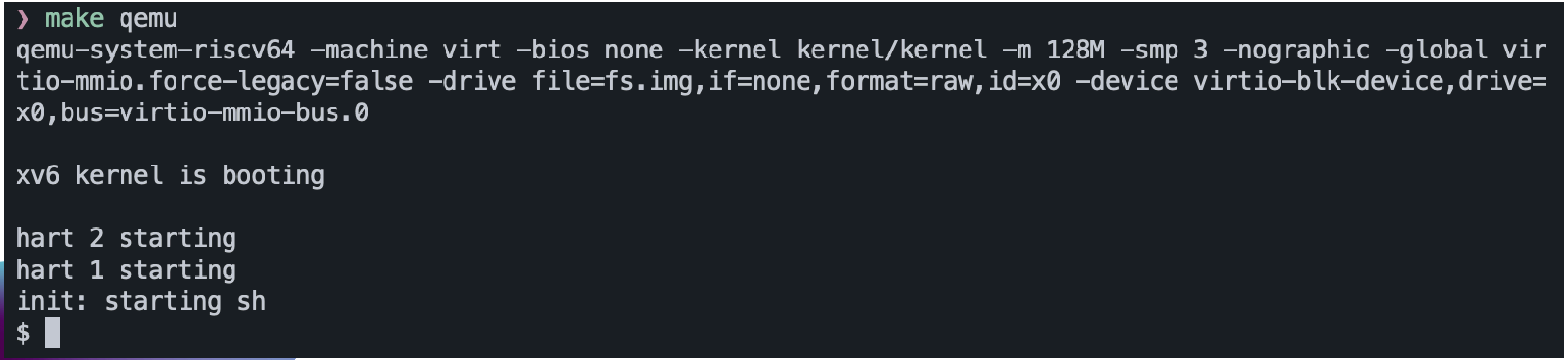
Try using hotspot if it doesn't download on IIIT network.

Also test your installation, as mentioned in the bottom of the page

# Running XV6

After installing, run make qemu in the directory where you cloned xv6.

You should see an output like this (with more lines before xv6 kernel is booting if its your first time running make qemu):



```
❯ make qemu
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -nographic -global vir
tio-mmio.force-legacy=false -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-device,drive=
x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ 
```

# Making a User Program

1. Create a C file for the actual function (if needed).
2. Edit the makefile -- update UPROGS

# Making a Syscall

1. Define a mapping for your syscall in syscall.h
2. Add your function prototype and array entry for the same thing in syscall.c (see how other syscalls have been added)
3. Define the function for your syscall in sysproc.c
4. Make whatever changes you need to make (depends on the syscall you are adding)

**this is just an outline so that you have a rough idea of what to do**

(some) other important files: proc.h, proc.c, trap.c, defs.h

## Some resources

https://github.com/YehudaShapira/xv6-explained/blob/master/Explanations.md

xv6 book (for reference):
https://pdos.csail.mit.edu/6.828/2023/xv6/book-riscv-rev3.pdf

https://www.youtube.com/playlist?list=PLbtzT1TYeoMhTPzyTZboW_j7TPAnjv9XB

# The types of sockets

**Stream sockets**

`SOCK_STREAM`

Use TCP
Reliable, errorless, sequential, two-way connections

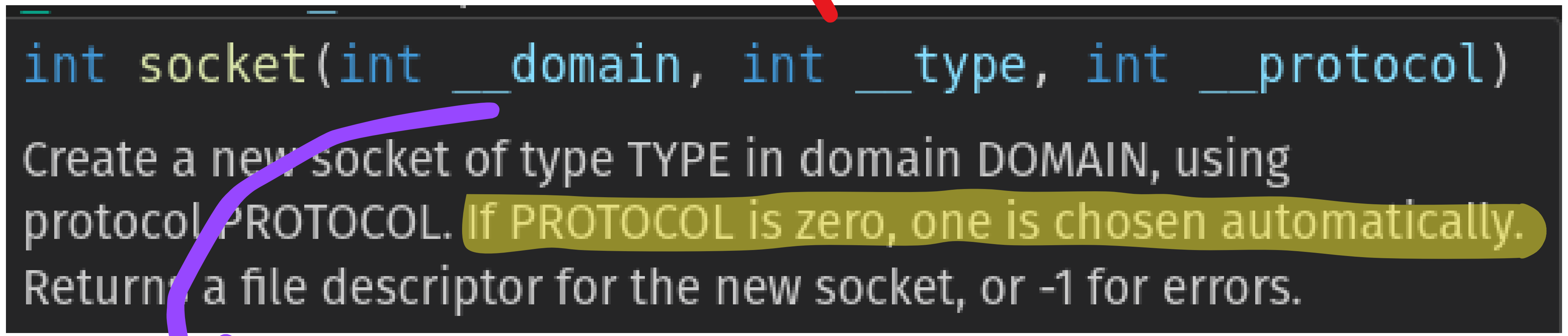
send(), recv()

**Datagram sockets**

`SOCK_DGRAM`

Use UDP
Less reliable, more prone to errors, faster, connectionless

sendto(), recvfrom()



```
int socket(int __domain, int __type, int __protocol)
```

Create a new socket of type TYPE in domain DOMAIN, using protocol PROTOCOL. If PROTOCOL is zero, one is chosen automatically. Returns a file descriptor for the new socket, or -1 for errors.

**AF_INET** for communication using IPv4 protocols (AF = Address Family)

https://beej.us/guide/bgnet/

## Blocking and non-blocking sockets

“Does the socket wait till the data has been sent/ received to return control to the program?”

Non-blocking sockets provide a way to keep sending and receiving packets without needing to wait for the step to complete

To use them: see fcntl(), SOCK_NONBLOCK (only one is needed)

https://www.scottklement.com/rpg/socktut/nonblocking.html

# Thank you

(and all the best)