xv6 setup guide

Follow the instructions given below for xv6 installation.
Run the following commands to get the xv6 source (if you are using lab machine)

- `wget https://www.cse.iitb.ac.in/~puru/courses/xv6-public.tar.gz`
- `tar -xf xv6-public.tar.gz`
- `cd xv6-public`
- `make`

If you are using a Linux environment on a personal machine, you will need a set of other tools as well for xv6 … use the following commands to install required packages.

- `sudo apt-get update`
- `sudo apt -y install build-essential gdb coreutils util-linux sysstat procps wget tar qemu`

The booklet describing/listing all source files is available **here**. (also after make in the xv6 dir)

xv6 runs on an x86 emulator called QEMU that emulates x86 hardware on your local machine.
In the xv6 folder, run the following command sequence to boot xv6 on an emulated machine.
QEMU boots the machine and if all goes well drops to a user space shell program.

- `make clean`
- `make qemu`
  Build everything and start qemu with the VGA console in a new window and the serial console in your terminal. To exit, either close the VGA window or press Ctrl-c or Ctrl-a x in your terminal.

- `make qemu-nox`
  Like make qemu, but run with only the serial console. To exit, press Ctrl-a x. This is particularly useful over SSH connections.

  Ctrl+A X ⇒ First press Ctrl + A (A is just key a, not the alt key), 2. then release the keys and press X.
- At the shell start with **ls** to list available programs and then execute a few of them.
- Look up the implementation of these programs. For example, cat.c is the source code for the cat program. Execute and lookup the following: ls, cat, wc, echo, grep etc. Understand how the syntax in some places is different from normal C syntax.
- Check the makefile to see how the program wc is set up for compilation.

FAQ

Q. make qemu-nox throws this error: "Couldn't find a working QEMU executable"

Ans. Sometimes, the default qemu package does not configure itself properly. Try installing the complete set of binaries for x86:

```
sudo apt install qemu-system-x86
```

and retry. Alternatively, if you are running Ubuntu directly on your machine (not on a VM/hypervisor), you can also install qemu-kvm

```
sudo apt install qemu-kvm
```

Please do not modify xv6-public's source by yourself, this can cause problems.

Q. make qemu-nox throws this error: "error: writing 1 byte into a region of size 0 [-Werror=stringop-overflow=]"

Ans. Edit makefile and add : `CFLAGS += -Wno-stringop-overflow` (link)

System calls related

To understand and work with system calls and process related information and actions, the following files of the xv6 OS are important —
`usys.S, user.h, defs.h, sysproc.c, syscall.h, syscall.c, proc.h, proc.c`

- **user.h** contains the xv6 system call declarations
- **usys.S** contains a list of system calls exported by the kernel and the corresponding invocation of the trap instruction
- **syscall.h** contains the mapping of the system call name to the system call number
- **syscall.c** contains helper functions to handle the system call entry, parse arguments, and pointers to the actual system call implementations
- **sysproc.c** contains the implementations of process-related system calls
- **defs.h** is a header file with function declarations in the xv6 kernel
- **proc.h** contains the process abstraction-related variable definitions
- **proc.c** contains implementations of various process-related system calls, functions and the scheduler, and also contains the declaration of ptable and several examples of functions traversing/using the process list
- System call-related functions are also listed in **sysfile.c**

All or most of these files will have to be used/updated to implement new system calls.
New files, new programs, and new data files need to be added to xv6 via the xv6 Makefile.
All changes are to be followed by a clean compile and build, followed by executing xv6.

Note that xv6 itself does not have a text editor or compiler support, so all xv6 source code changes are on the host machine then **xv6.img** and **fs.img** are used as inputs to QEMU.