

1) This nsh shell is meant to work on xv6 OS only, however the x86 xv6 is deprecated. So I used the RISC-V one, which is a bit tedious to set up, but is nevertheless easy. This is where you find the RISC-V version: <https://github.com/mit-pdos/xv6-riscv/>

You can `git clone` it into any location.

2) First install the RISC-V GCC toolchain (from this link): <https://github.com/stnolting/riscv-gcc-prebuilt/releases>

Ubuntu 20.04 LTS @ x64: gcc-12.1.0, rv32i, ilp32, newlib

Build on: Ubuntu 20.04 LTS, 64-bit x86 machine (actually, Ubuntu on Windows)

Toolchain prefix: `riscv32-unknown-elf`

Tool chain archive	gcc	binutils	march	mabi	clib
<code>riscv32-unknown-elf.gcc-12.1.0.tar.gz</code>	<code>12.1.0</code>	<code>2.39</code>	<code>rv32i</code>	<code>ilp32</code>	<code>newlib</code>

Assets

3

`riscv32-unknown-elf.gcc-12.1.0.tar.gz`500 MBAug 20, 2022

`Source code (zip)`Aug 19, 2022

`Source code (tar.gz)`Aug 19, 2022

Ubuntu 20.04 LTS @ x64: gcc-10.2.0, rv32i, ilp32, newlib

2) Also install QEMU RISC-V emulator.

using command `sudo apt install qemu-system-riscv64`

```
user/usys.S \
user/_cat user/_echo user/_forktest user/_grep user/_init user/_kill user/_ln user/_ls user/_mkdir user/_rm user/_sh user/_stressfs user/_usertests user/_grind user/_wc
[sandstorm@Victus16 xv6-riscv]$ sudo apt install qemu-system-riscv64
[sudo] password for sandstorm:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'qemu-system-misc' instead of 'qemu-system-riscv64'
The following packages were automatically installed and are no longer required:
  iputils-arping ncurses-term openssh-sftp-server ssh-import-id
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  samba vde2
The following NEW packages will be installed:
  qemu-system-misc
0 upgraded, 1 newly installed, 0 to remove and 48 not upgraded.
Need to get 41.0 MB of archives.
After this operation, 185 MB of additional disk space will be used.
Ign:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 qemu-system-misc amd64 1:6.2+dfsg-2ubuntu6.6
Ign:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 qemu-system-misc amd64 1:6.2+dfsg-2ubuntu6.6
Ign:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 qemu-system-misc amd64 1:6.2+dfsg-2ubuntu6.6
Ign:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 qemu-system-misc amd64 1:6.2+dfsg-2ubuntu6.6
Get:1 http://security.ubuntu.com/ubuntu jammy-updates/main amd64 qemu-system-misc amd64 1:6.2+dfsg-2ubuntu6.6 [41.0 MB]
18% [1 qemu-system-misc 9,193 kB/41.0 MB 22%]
```

Now you have all the required pre-requisites.

3) Paste the nsh.c and Makefile (or alternatively just clone this repo if you're fine with working in an older xv6 version).

4) Go inside the folder and open the terminal in the location where Makefile is located.

5) Comment out the “\$K/console.o” first and run make.

6) Run `make fs.img`

7) Run `make qemu`

8) Uncomment out the aforementioned line and rerun the three commands.

9) After that you should be running the QEMU emulator, running the version of RISC-V version of xv6.

(The output should be something similar to)

```
sandstorm@Victus16:~/programming/C_projects/nsh/xv6-riscv
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-plt -no-pie -c -o user/wc.o user/wc.c
riscv64-linux-gnu-ld -z max-page-size=4096 -T user/user.ld -o user/wc user/wc.o user/lib.o user/usys.o user/printf.o user/unalloc.o
riscv64-linux-gnu-objdump -S user/wc > user/wc.asm
riscv64-linux-gnu-objdump -t user/wc | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/wc.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-plt -no-pie -c -o user/zombie.o user/zombie.c
riscv64-linux-gnu-ld -z max-page-size=4096 -T user/user.ld -o user/zombie user/zombie.o user/lib.o user/usys.o user/printf.o user/unalloc.o
riscv64-linux-gnu-objdump -S user/zombie > user/zombie.asm
riscv64-linux-gnu-objdump -t user/zombie | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/zombie.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-plt -no-pie -c -o user/nsh.o user/nsh.c
riscv64-linux-gnu-ld -z max-page-size=4096 -T user/user.ld -o user/nsh user/nsh.o user/lib.o user/usys.o user/printf.o user/unalloc.o
riscv64-linux-gnu-objdump -S user/nsh > user/nsh.asm
riscv64-linux-gnu-objdump -t user/nsh | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/nsh.sym
mkfs/mkfs fs.img README user/_cat user/_echo user/_forktest user/_grep user/_lnt user/_kill user/_ln user/_ls user/_mkdir user/_rm user/_sh user/_stressfs user/_usertests user/_grind user/_wc user/_zombie user/_nsh
mmeta 46 (boot, super, log blocks 30 inode blocks 13, bitnap blocks 1) blocks 1954 total 2000
balloc: first 793 blocks have been allocated
balloc: write bitnap block at sector 45
sandstorm@Victus16:~/programming/C_projects/nsh/xv6-riscv$ make qemu
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -nographic -global virtio-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 1 starting
hart 2 starting
init: starting sh
$ nsh
0 ls
. 1 1 1024
. 1 1 1024
README 2 2 2305
cat 2 3 32856
echo 2 4 31648
forktest 2 5 13808
grep 2 6 30224
lnt 2 7 32184
kill 2 8 31648
ln 2 9 31472
ls 2 10 34770
mkdir 2 11 31720
rm 2 12 31704
sh 2 13 54090
stressfs 2 14 32520
usertests 2 15 188472
grind 2 16 47528
wc 2 17 33784
zombie 2 18 31064
nsh 2 19 47008
console 3 20 0
$ cat README | grep code
The code in the files that constitute xv6 is
0
```

You can see the shell works just fine :)