xv6 Hands-on Session

CS744 Design and Engineering of Computing Systems

Autumn 2024

about xv6

xv6? a simple, Unix-like teaching operating system

Learn main concepts of operating systems by studying an example kernel - xv6

- xv6 is based on Unix Version (v6).
- implemented in ANSI C.
- two versions, one for x86 hardware and one for RISC-V hardware

  this hands-on – based on x86 version

The job of an operating system

- share a computer among multiple programs
- provide a more useful set of services than the hardware alone supports.

where to run xv6?

OS runs on (real) hardware.

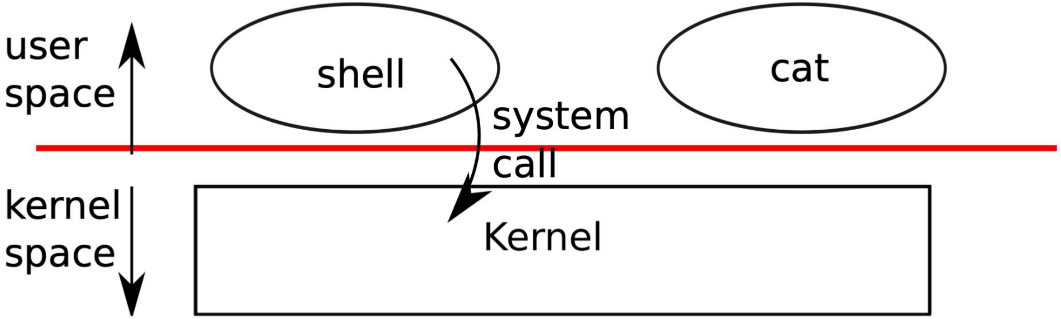
xv6 runs on a hardware emulator called QEMU.

benefits

- run xv6 in any machine (ARM, x86, RISC, etc.)
- kernel crashes can be handled gracefully.
- etc.

let’s get started

```
tauser@sl2-1:~/ricky$ wget https://www.cse.iitb.ac.in/~puru/courses/xv6-public.tar.gz
tauser@sl2-1:~/ricky$ tar -xf xv6-public.tar.gz
tauser@sl2-1:~/ricky$ cd xv6-public/
tauser@sl2-1:~/ricky/xv6-public$ ls
```



let’s get started

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.

```
user@linux:~/ricky$ sudo apt-get update
user@linux:~/ricky$ sudo apt -y install build-essential gdb coreutils util-linux
sysstat procps wget tar qemu
```

user-mode code

- user.h — declarations of system call wrappers and standard library functions
- usys.S — assembly code (generated by preprocessor macros) for system call wrappers
- ulib.c, printf.c, umalloc.c — user mode standard library, including printf, malloc, free, …
- supplied xv6 programs
  - cat.c, echo.c, forktest.c, grep.c, init.c, kill.c, ln.c, ls.c, mkdir.c, rm.c, sh.c, shutdown.c, stressfs.c, usertests.c, wc.c, zombie.c

shared user/kernel header and utility files

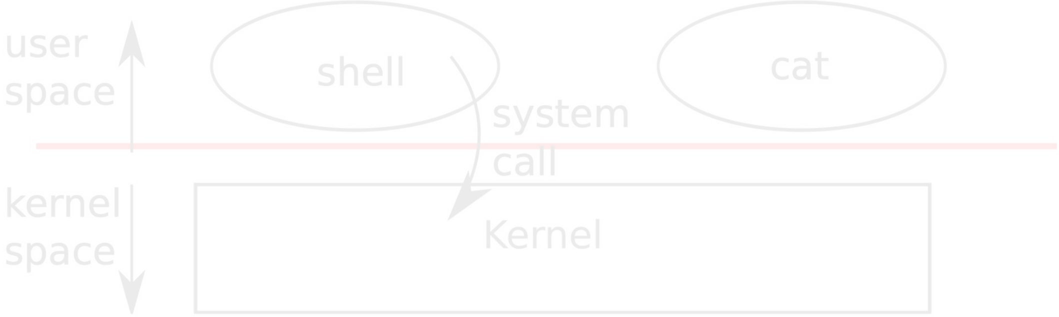
- `types.h`, `fcntl.h`, `stat.h`

utility (non-xv6) programs

- `mkfs.c` — create filesystem images so xv6 can boot in qemu

let's get started

```
tauser@sl2-1:~/ricky$ wget https://www.cse.iitb.ac.in/~puru/courses/xv6-public.tar.gz
tauser@sl2-1:~/ricky$ tar -xf xv6-public.tar.gz
tauser@sl2-1:~/ricky$ cd xv6-public/
tauser@sl2-1:~/ricky/xv6-public$ ls
```



kernel-mode code: everything else

- `defs.h` — declarations of functions callable within the kernel
- `param.h` — declarations of hard-coded limits (like number of file descriptors per process)
- process-related:
  - `proc.h`, `proc.c`
  - `exec.c`, `elf.h` — loading executables into memory
  - `file.h`, `file.c`, `pipe.c`, — file/file descriptor handling related code
- memory management:
  - `mmu.h`, `vm.c`, `kalloc.c`
- multicore
  - `mp.c`, `mp.h`
- synchronization
  - `spinlock.h`, `spinlock.c`, `sleeplock.h`, `sleeplock.c`
- exception/trap handling:
  - `traps.h`, `trap.c, trapasm.S, ioapic.c, lapic.c, picirq.c`

kernel-mode code: everything else (continued)

- system call handling
  - `syscall.h`, `syscall.c` — system call handling/dispatch code
  - `sysproc.c` — process-related system call implementations
  - `sysfile.c` — file-related system call implementations
  - `kshutdown.c`,
- I/O
  - `buf.h, bio.c, console.c, ide.c, kbd.h, kbd.c, memide.c, uart.c`
- filesystem:
  - `fs.h`, `fs.c`, `log.c`,
- boot handling
  - `bootmain.c`, `bootasm.S`, `main.c`

let’s get started

```
tauser@sl2-1:~/ricky$ wget https://www.cse.iitb.ac.in/~puru/courses/xv6-public.tar.gz
tauser@sl2-1:~/ricky$ tar -xf xv6-public.tar.gz
tauser@sl2-1:~/ricky$ cd xv6-public/
tauser@sl2-1:~/ricky/xv6-public$ make
```

What is make doing?

let’s get started

```
tauser@sl2-1:~/ricky$ wget https://www.cse.iitb.ac.in/~puru/courses/xv6-public.tar.gz
tauser@sl2-1:~/ricky$ tar -xf xv6-public.tar.gz
tauser@sl2-1:~/ricky$ cd xv6-public/
tauser@sl2-1:~/ricky/xv6-public$ make
```

What is make doing?

Compiling!!

How to boot into xv6?

```
tauser@sl2-1:~/ricky/xv6-public$ make qemu-nox
```

After bootup, xv6 creates a init program which opens a shell in which common commands and other user programs can be run.

```
init: starting sh
$
```

How to boot into xv6?

```
tauser@sl2-1:~/ricky/xv6-public$ make qemu-nox
```

After bootup, xv6 creates a init program which opens a shell in which common commands and other user programs can be run.

```
init: starting sh
$ ls
```

How to boot into xv6?

```
tauser@sl2-1:~/ricky/xv6-public$ make qemu-nox
```

After bootup, xv6 creates a init program which opens a shell in which common commands and other user programs can be run.

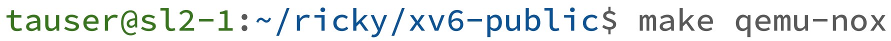
How to boot into xv6?

```
tauser@sl2-1:~/ricky/xv6-public$ make qemu-nox
```

After bootup, xv6 creates a init program which opens a shell in which common commands and other user programs can be run.

```
init: starting sh
$ ls
.              1 1 512
..             1 1 512
README         2 2 2286
cat            2 3 15464
echo           2 4 14348
forktest       2 5 8792
grep           2 6 18308
init           2 7 14968
kill           2 8 14432
ln             2 9 14328
ls             2 10 16896
mkdir          2 11 14456
rm             2 12 14436
sh             2 13 28492
stressfs       2 14 15364
usertests      2 15 62864
wc             2 16 15892
zombie         2 17 14012
console        3 18 0
```

How to boot into xv6?



```
tauser@sl2-1:~/ricky/xv6-public$ make qemu-nox
```

After bootup, xv6 creates a init program which opens a shell in which common commands and other user programs can be run.

Let’s exit from xv6 -

Ctrl+A X

1. First press Ctrl + A (A is just key a, not the alt key),
2. then release the keys and press X

init and sh

After bootup, xv6 creates a **init** program which opens a **shell** in which common commands and other user programs can be run.

See contents of `init.c` and `sh.c`

Makefile - one tool to rule them all

What is required for xv6 to run?

Makefile - one tool to rule them all

What is required for xv6 to run?

# Makefile Syntax & Crux of Makefile:

- A makefile consists of set of rules. A rule looks like:

```
targets: prerequisites
	command
	command
	command
```

Makefile - one tool to rule them all

What is required for xv6 to run?

See prerequisites of qemu-nox target!

Makefile - one tool to rule them all

What is required for xv6 to run?

See prerequisites of qemu-nox target!

```
fs.img xv6.img
```

Makefile - one tool to rule them all

Important (for you) targets of xv6 Makefile:

UPROGS - Makefile variable which lists names of all user programs which are available after xv6 boot up.

`fs.img`: List of files to be added to the xv6 startup disk (imagefile).

```
fs.img: mkfs README $(UPROGS)
    ./mkfs fs.img README $(UPROGS)
```

Task 1 - Add a new text file in xv6 environment

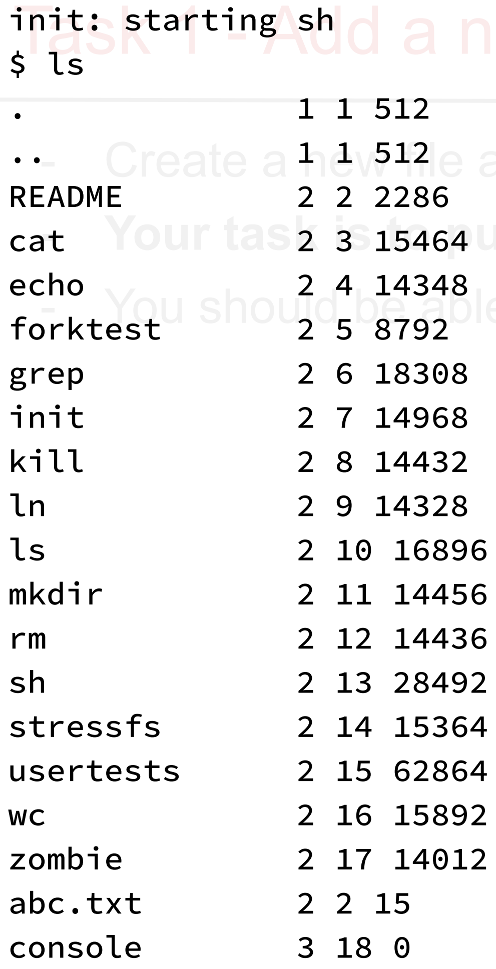
- Create a new file abc.txt with contents “I am ironman!”
- **Your task is to put the file inside the xv6 file system.**
- You should be able to boot into xv6, find the abc.txt and cat the file.

Task 1 - Add a new text file in xv6 environment

- Create a new file abc.txt with contents “I am ironman!”

Your task is to put the file inside the xv6 file system.

- You should be able to boot into xv6, find the abc.txt and cat the file.



```
init: starting sh
$ ls
.              1 1 512
..             1 1 512
README         2 2 2286
cat            2 3 15464
echo           2 4 14348
forktest       2 5 8792
grep           2 6 18308
init           2 7 14968
kill           2 8 14432
ln             2 9 14328
ls             2 10 16896
mkdir          2 11 14456
rm             2 12 14436
sh             2 13 28492
stressfs       2 14 15364
usertests      2 15 62864
wc             2 16 15892
zombie         2 17 14012
abc.txt        2 2 15
console        3 18 0
```

Task 1 - Add a new text file in xv6 environment

- Create a new file abc.txt with contents “I am ironman!”
- **Your task is to put the file inside the xv6 file system.**
- You should be able to boot into xv6, find the abc.txt and cat the file.

```
$ cat abc.txt
I am ironman!!
```

Task 2 - Add a new userspace program in xv6 environment

- Create a new userspace program hw.c which should print “Hello CS744”.
- **Your task is to put the file and its compiled userspace program inside the xv6 file system.**
- You should be able to boot into xv6, find the `hw.c`, cat the file and run the executable hw.
- To get started look into `user.h` and `types.h`.

Task 2 - Add a new userspace program in xv6 environment

- Create a new userspace program hw.c which should print “Hello CS744”
- **Your task is to put the file and its compiled userspace program inside the xv6 file system.**

```
$ cat hw.c
#include "types.h"
#include "user.h"

int main()
{
  printf(1, "Hello World\n");
  exit();
}
```

xv6 system calls

Syscall listing - can also be found in `user.h`

`fork()`, `exec()`, `wait()`, `getpid()`, `kill()`, `pipe()`, `read()`, `write()`, `open()`, `close()` etc.

Task 3 - Use system call in your userspace program

- Just like lab2’s 2a task implement a version of the cat command (name it `mycat.c`) using the **fork** system call to create the child process that reads contents from **STDIN** and writes them to **STDOUT** using system calls read and write **(NOT printf and scanf)**. It should read from standard input (STDIN) till a new line character and output to standard output (STDOUT).

```
$ mycat
>>> OS is critical for world peace!
OS is critical for world peace!
>>>
```

What’s next?

- Implementing your own system call.
- Adding your own memory management ideas.
- Adding networking support.
- Adding pseudo file system.
- Multi threading support.
- Your imagination…..