CSE 3025 Principles of Operating Systems
Fall 2025
Lab 1 xv6 and Utilities

This project is to be completed individually. Due by the end of Tuesday, Sept. 2, 2025.

Boot xv6

Please check the lab tools page on Canvas for information about how to set up your computer to run the labs.

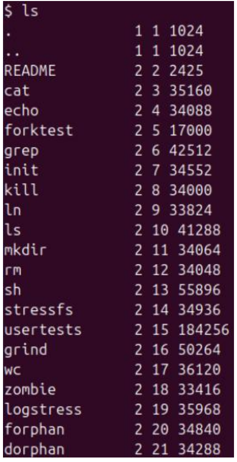
First, clone a copy of xv6 for lab 1:

```
$mkdir xv6lab1
$cd xv6lab1
$git clone https://github.com/mit-pdos/xv6-riscv.git
```

Then, build and run xv6:

```
$cd xv6-riscv
$make qemu
```

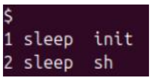
If you type `ls` at the prompt, you should see output similar to the following:



```
$ ls
.              1 1 1024
..             1 1 1024
README         2 2 2425
cat            2 3 35160
echo           2 4 34088
forktest       2 5 17000
grep           2 6 42512
init           2 7 34552
kill           2 8 34000
ln             2 9 33824
ls             2 10 41288
mkdir          2 11 34064
rm             2 12 34048
sh             2 13 55896
stressfs       2 14 34936
usertests      2 15 184256
grind          2 16 50264
wc             2 17 36120
zombie         2 18 33416
logstress      2 19 35968
forphan        2 20 34840
dorphan        2 21 34288
```

These are the files that `mkfs` includes in the initial file system; most are programs you can run. You just ran one of them: `ls`.

xv6 has no `ps` command, but, if you type Ctrl-p, the kernel will print information about each process. If you try it now, you'll see two lines: one for `init`, and one for **sh**.



```
$
1 sleep  init
2 sleep  sh
```

To quit qemu, type: `Ctrl-a x` (press `Ctrl` and `a` at the same time, followed by `x`).

Exercise 1: Add One User-level Program (New Shell Command) on xv6 (10 pts)
Add a user-level program `myapp.c` on xv6 that print out a “`Hello World!`” message.

Adding the user-level program `myapp.c` in user space of xv6 requires two actions:

(1) Create the file `myapp.c`, under the directory `/user`.

(2) In `Makefile`, add `_myapp` to the parameter `UPROGS`.

Some hints:

(1) A xv6 user-level program does not use typical C header files. Please check existing user-level programs (e.g., `wc.c`) under `/user` directory on the header files.

(2) User programs on xv6 have a limited set of library functions available to them. You can see the list in `user/user.h`; the source (other than for system calls) is in `user/ulib.c`, `user/printf.c`, and `user/umalloc.c`.

(3) A user-level program must end with exit(0)

Exercise 2: Add New xv6 Command (30 pts)

In Linux and xv6, by default, the `wc` command shows the line count, word count, and character count of list of files:

```
$ wc README
46 319 2292 README
```

However, in Linux, you can provide the following command line options:

```
wc [-l] [-w] [-c]
```

Using these options can control the output of the `wc` command:

```
-l lines
-w words
-c characters
```

By default without options, wc prints all three of these values.

```
$ wc -l README
46 README
```

For this part of the lab you need to create a new shell command (`wc_new`) which modifies the existing implementation of `wc` in xv6 to support these three new options.

Note that `wc` can also take multiple files as arguments:

```
$ wc README domains.txt
46 319 2292 README
4 8 106 domains.txt
```

The options you add should still work with multiple files:

```
$ wc -l README domains.txt
46 README
4 domains.txt
```

Finally, the `wc` command can also take input from `stdin`, your new command should also be able to work with `stdin`:

```
$ cat README | wc -l
46
```

Note that the options can be provided in any order.

Exercise 3: Text Message (50 pts)

Write a user-level program that uses xv6 system calls to send a string *txtmsg* specified by the user between two processes over a pair of pipes, one for each direction. The parent should send *txtmsg* to the child; the child should print "<pid>: received txtmsg", where <pid> is its process ID, write txtmsg on the pipe to the parent, and exit; the parent should read the string from the child, print "<pid>: received txtmsg", and exit. Your solution should be in the file `user/txtmsg.c`.

Some hints:

(1) Add the program to `UPROGS` in Makefile.

(2) You'll need to use the `pipe`, `fork`, `write`, `read`, and `getpid` system calls.

(3) Run the program from the xv6 shell and it should produce the following output:

```
 $make qemu-nox
 ...
 init: starting sh
 $ txtmsg hello!
 4: received hello!
 3: received hello!
 $
```

Your program should exchange a string between two processes and produce output as shown above.

Time Logging (10 pts)

As you work on each of the programming labs, keep a detailed record of how you spend your time working on the lab. For a group-based lab, EACH team member must keep a separate log of time spent on the lab. To do this, place a log file with a name of the form '`labN-firstname-lastname.txt`' in the top-level lab source directory. The log file should have the following general form:

```
ESTIMATE of time to complete lab: 15 hours

      Start  Time  Lab
Date  Time   Spent Part Work completed
----  -----  ----  ---- ---------
2/08  10:15  1:00  1    Read assignment, completed cprintf exercise
2/09  20:15  2:00  1    Studied IA-32 programmer's guide, got backtrace working
2/10  12:45  0:30  2    Reading IA-32 system guide on trap handling
2/10  14:00  0:30  2    Discussing trap handling approach with teammates
2/10  16:00  2:00  2    Helped debug Bob's implementation of _alltraps
2/12  21:20  2:00  4    First cut on coding up physical page allocation
2/13  09:00  3:00  3    Meet, help teammates debug protected control transfer
2/13  20:00  5:00  4    Mysterious memory corruption bug in my page allocator
2/14  10:00  1:00  4    Aha!  Was overwriting the last page of the kernel.
              -----
              17:00  TOTAL time spent
a brief discussion of the major difficulties encountered
```

The time log contains:

- An estimate of the time you think will be required to complete the lab, made prior to writing any code. This estimate must be on the *only* line in the log file containing the string ESTIMATE in all-caps.
- The total time you actually spent in the lab. This time must be on the *only* line in the log file containing the string TOTAL in all-caps.
- A brief discussion (100 words *minimum* of the major difficulties you encountered while writing and debugging the code (there should always be some).

What to submit

In `lab1-firstname-lastname.txt`, please include your time logging for this lab (10 points). AFTER committing all changes, you run `make clean`, and `tar zcf ../lab1-firstname-lastname.tar.gz ../xv6lab1` assuming your xv6 source directory is `xv6lab1`. Then upload the complete tarred and compressed xv6 directory (`lab1-firstname-lastname.tar.gz`) through the submission link on Canvas.