Homework: xv6 system calls

Submit your solutions before the beginning of the next lecture to the submission web site.

You will modify xv6 to add a system call. You can use the same setup as for the boot homework.

Part One: System call tracing

Your first task is to modify the xv6 kernel to print out a line for each system call invocation. It is enough to print the name of the system call and the return value; you don't need to print the system call arguments.

When you're done, you should see output like this when booting xv6:

```
...
fork -> 2
exec -> 0
open -> 3
close -> 0
$write -> 1
 write -> 1
```

That's init forking and execing sh, sh making sure only two file descriptors are open, and sh writing the $ prompt. (Note: the output of the shell and the system call trace are intermixed, because the shell uses the write syscall to print its output.)

Hint: modify the syscall() function in syscall.c.

Optional challenge: print the system call arguments.

Part Two: Date system call

Your second task is to add a new system call to xv6. The main point of the exercise is for you to see some of the different pieces of the system call machinery. Your new system call will get the current UTC time and return it to the user program. You may want to use the helper function, `cmostime()` (defined in `lapic.c`), to read the real time clock. `date.h` contains the definition of the `struct rtcdate` struct, which you will provide as an argument to `cmostime()` as a pointer.

You should create a user-level program that calls your new date system call; here's some source you should put in `date.c`:

```
#include "types.h"
#include "user.h"
#include "date.h"

int
main(int argc, char *argv[])
{
  struct rtcdate r;

  if (date(&r)) {
    printf(2, "date failed\n");
```

```
    exit();
  }

  // your code to print the time in any format you like...

  exit();
}
```

In order to make your new `date` program available to run from the xv6 shell, add `_date` to the `UPROGS` definition in `Makefile`.

Your strategy for making a date system call should be to clone all of the pieces of code that are specific to some existing system call, for example the "uptime" system call. You should grep for uptime in all the source files, using `grep -n uptime *.[chS]`.

When you're done, typing `date` to an xv6 shell prompt should print the current UTC time.

Write down a few words of explanation for each of the files you had to modify in the process of creating your date system call.

Optional challenge: add a dup2() system call and modify the shell to use it.

> **Submit**: Your explanations of the modifications for date in a file named hwN.txt where N is the homework number as posted on the schedule.