Homework #6 Solution

Question 1 (6 pt.)

Add a new system call to xv6 to obtain the system date and time. The interface for this system call is the following:

```
struct rtcdate
{
        uint second;
        uint minute;
        uint hour;
        uint day;
        uint month;
        uint year;
};

int date(struct rtcdate *d);
```

The system call takes a structure of type `struct rtcdate` (already defined in header file `date.h`) as its first argument passed by reference, and always returns 0. You will need to understand the content of the following files, and modify them accordingly:

- `syscall.c`. Add the declaration of function `date()`. Since the system call body reads the arguments from the user-space stack, the system call prototype at this level should take no arguments (`void`). You also need to add the system call to the `syscalls` array.
- `syscall.h`. Add a unique numeric identifier for the new system call.
- `sysproc.c`. You can use this kernel-level source file to append the implementation of the system call. Again, notice that the body of the system call needs to grab the arguments from the user stack. Function `argint()` allows you to read one argument interpreted as a 32-bit integer value, and you can see an example of its invocation on the same file in the body of function `sys_sleep()`, among others. However, in our case, we are interested in interpreting the argument as a pointer, for which you can either cast the result of `argint()`, or directly invoke function `argptr()` instead. You can use kernel-level function `cmostime()`, defined in `lapic.c`, in order to query the system time using I/O operations.
- `user.h`. This is a user-level header file that should include the full prototype of the system call, as invoked by the user code.
- `usys.S`. This file contains assembly code that converts a user-level invocation to a system call function into a trap instruction with the proper system call code. You need to add a new macro invocation for the new system call, following the pattern observed for other system calls.

Modify these five files and pack them into a single ZIP file named `q1.zip`, and upload it on Blackboard as part of your submission. Do not add all the xv6 code, just those five files. When extracted into the xv6 directory, your code should compile and run properly on the Linux CoE machines by just typing `make qemu`. You can test it with the user program requested in next question.

Solution

File `syscall.c`

```
[...]
extern int sys_date(void);

static int (*syscalls[])(void) = {
[SYS_fork]    sys_fork,
    [...]
[SYS_date]    sys_date
};
```

File `syscall.h`

```
[...]
#define SYS_date   22
[...]
```

File `sysproc.c`

```
[...]
int sys_date(void)
{
        struct rtcdate *d;
        argint(0, (int *) &d);
        cmostime(d);
        return 0;
}
```

File `user.h`

```
[...]
int date(struct rtcdate *date);
[...]
```

File `usys.S`

```
[...]
SYSCALL(date)
[...]
```

Question 2 (4 pt.)

Create a new user program named `date.c` that will serve as an xv6 command-line tool to obtain the result of the `date()` system call. The program should be invoked from the xv6 shell without any arguments, and should provide the current date and time in a human-readable format, as follows:

```
$ date
5/4/2015 20:38:6
```

You can follow the directions in the support material in order to create and compile a new user-space program for xv6, as well as to make it available in the root directory of the xv6 file system. Upload your code in a file named `date.c`.

Solution

File `date.c`

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

int main(int argc, char **argv)
{
        struct rtcdate d;
        if (date(&d) < 0)
        {
                printf(1, "Error: cannot obtain date\n");
                exit();
        }

        // Print date
        printf(1, "%d/%d/%d %d:%d:%d\n",
                        d.day,
                        d.month,
                        d.year,
                        d.hour,
                        d.minute,
                        d.second);
        exit();
}
```