**xv6 system calls**

## 一、System call tracing

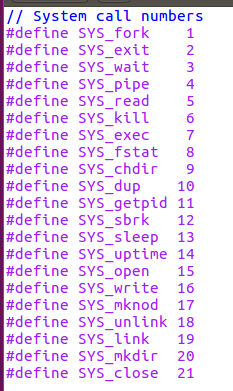
### 0.要求

修改xv6内核以为每个系统调用调用打印一行。只需要打印系统调用的名称和返回值。

### 1.实验过程

这一步我们要跟踪系统调用，并把他们打印在终端上。

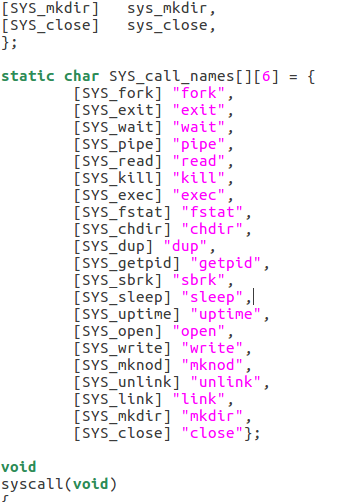
首先，系统调用的名称和对应的序号定义在syscall.h中，我们要看一下 syscall.h中的内容：



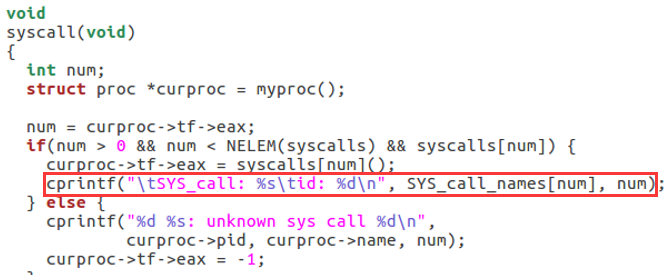
要在终端上显示的就是上面这些名字和对应的编号。

下面，打开文件syscall.c，开始进行文件修改。主要是在syscall函数内添加printf语句。

在文件前面添加编号和系统调用名称相互对应的数组：

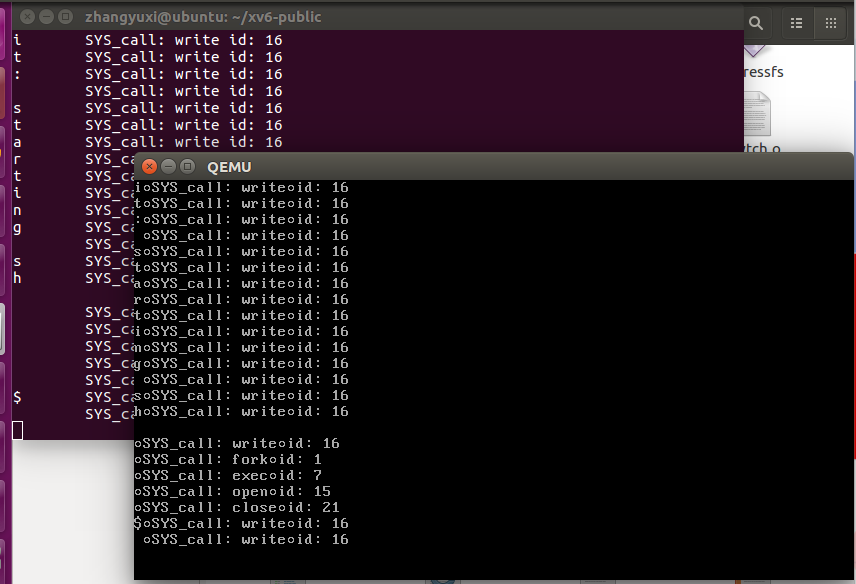


然后在函数syscall中添加对应的输出命令：



第一部分的代码修改完成。

在ubuntu虚拟机中用make qemu语句编译运行：



## 二、Date system call

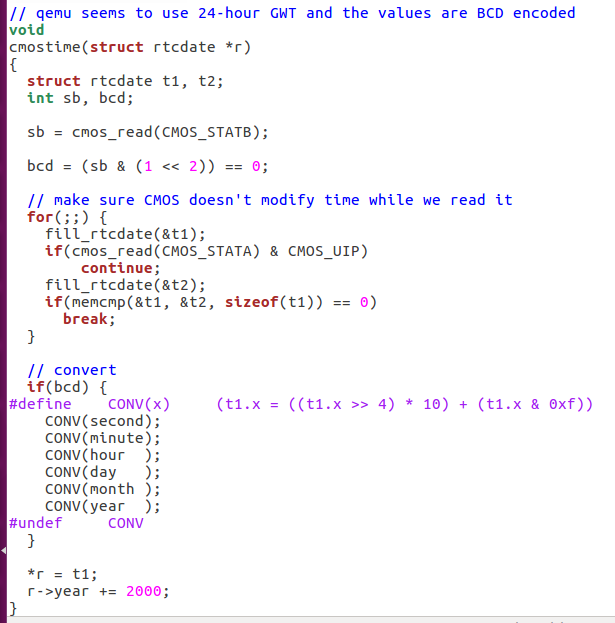
### 0.要求

向xv6添加一个新的系统调用。新系统调用将获取当前UTC时间，并将其返回给用户程序。可能需要使用辅助函数cmostime（） （在lapic.c中定义）来读取实时时钟。date.h 包含rtcdate struct结构的定义，你将提供它作为cmostime（）的参数。

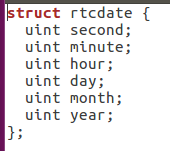
### 1.实验过程

我们要添加一个系统调用函数，使其可以返回当前系统时间。

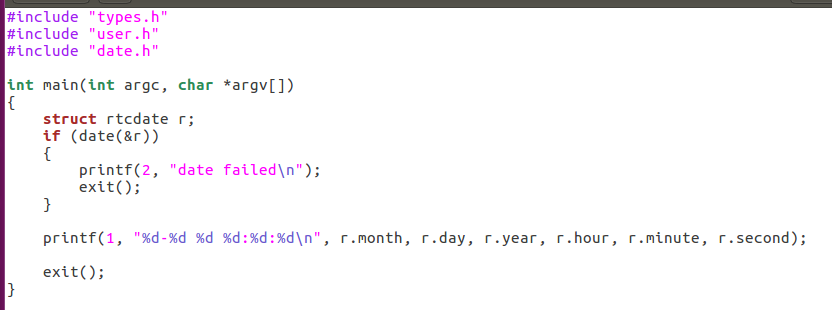
lapic.c 中定义了函数 cmostime() 这个函数用来读取时钟的时间。



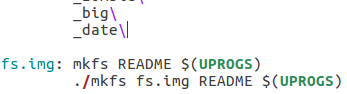
date.h 中定义了 rtcdate 这个结构体，它作为参数提供给函数 cmostime() 。



添加使用这个系统调用函数的方法：新建文件 date.c ，内容如下：

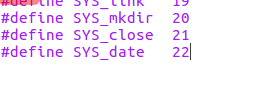


在 Makefile 中添加对应命令的定义：

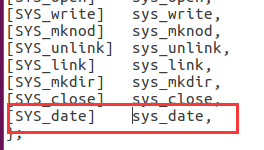
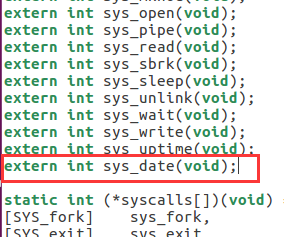


接下来添加系统调用函数 date()

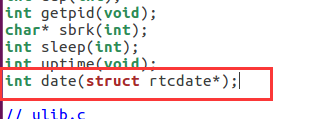
在 syscall.h 的最后一行添加系统调用编号：



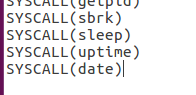
在 syscall.c 的添加系统调用函数的外部声明



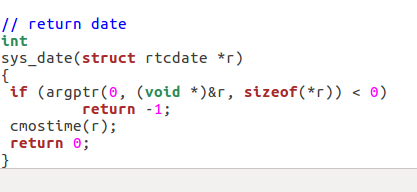
在 user.h 的添加用户态函数的定义



在 usys.S 的最后一行添加用户态函数的实现：

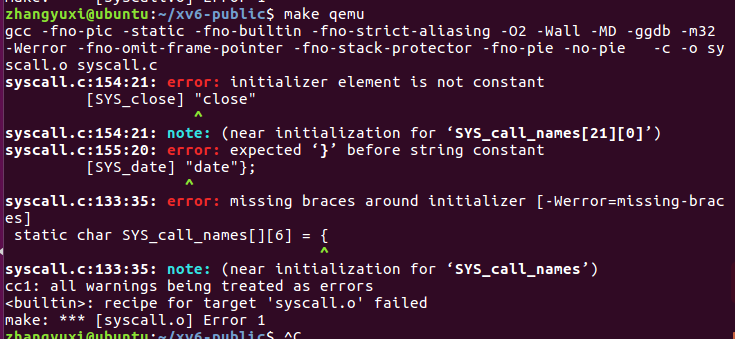


在 sysproc.c 的最后一行之后添加系统调用函数的实现

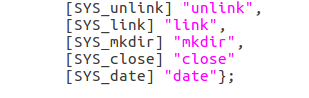


修改完成。

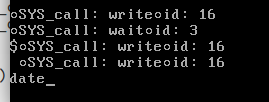
使用make qemu运行，发现syscall处报错。



检查之后发现，“close”后面少了一个逗号。

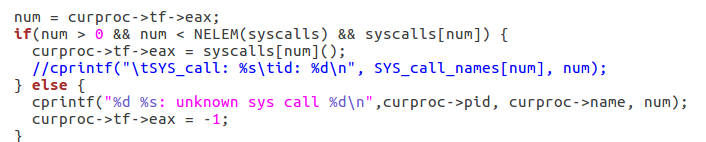


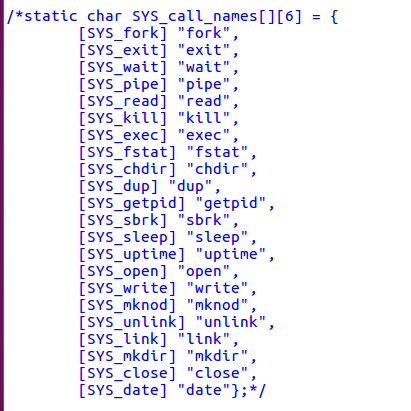
修正之后再次运行，输入date命令后发现，由于第一问的输出，找不到自己想要的日期输出：





为了看到第二问的输出，先将第一问在 syscall.c 中增加的代码注释掉：





再次运行，输入date，终端根据写好的时间格式输出了UTC时间。

