CS 202: Advanced Operating Systems

You can use sledge server for the course labs.

```
$ ssh -X username@sledge.cs.ucr.edu
```

To download xv6:

```
$ git clone https://github.com/mit-pdos/xv6-public.git
$ cd xv6-public
```

Links:

xv6 book
xv6 indexed/cross referenced code

To run XV6:

```
$ make qemu
```

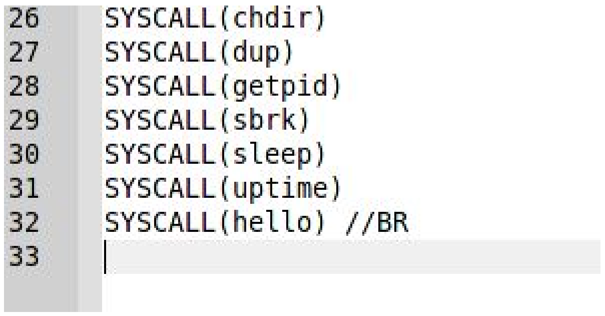
If the mouse pointer gets stuck in the QEMU emulator window press:
Ctrl + Alt + G

To create a system call:

Make a system call "sys_hello" that call a kernel function that displays:
"Hello from the kernel space!"

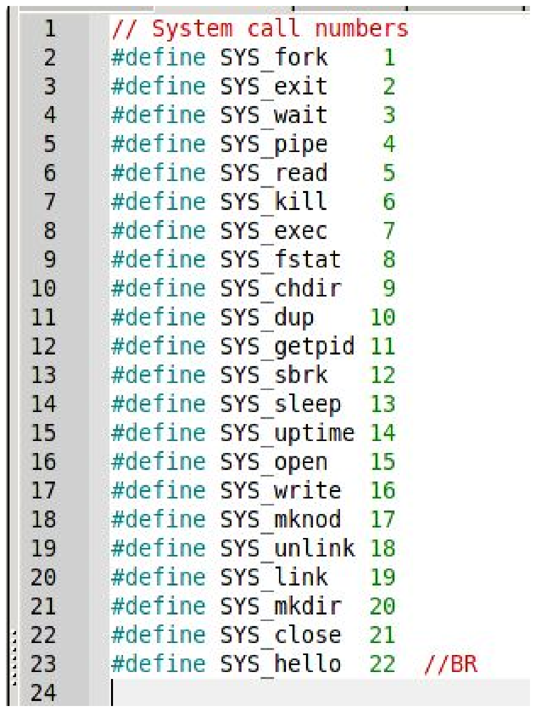
To do that open the following files and add the line(s) with //BR comment:

In "usys.S"



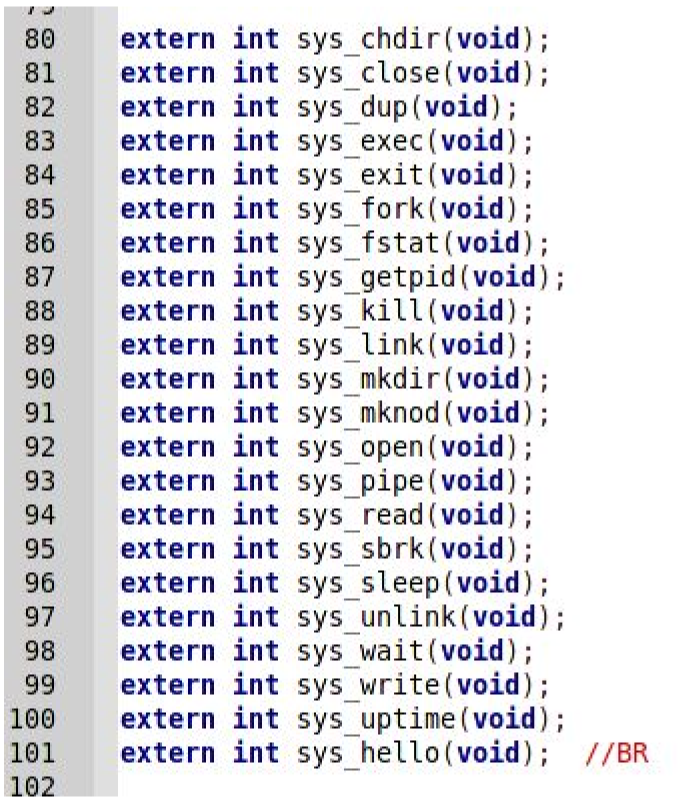
```
SYSCALL(chdir)
SYSCALL(dup)
SYSCALL(getpid)
SYSCALL(sbrk)
SYSCALL(sleep)
SYSCALL(uptime)
SYSCALL(hello) //BR

```

In “syscall.h”



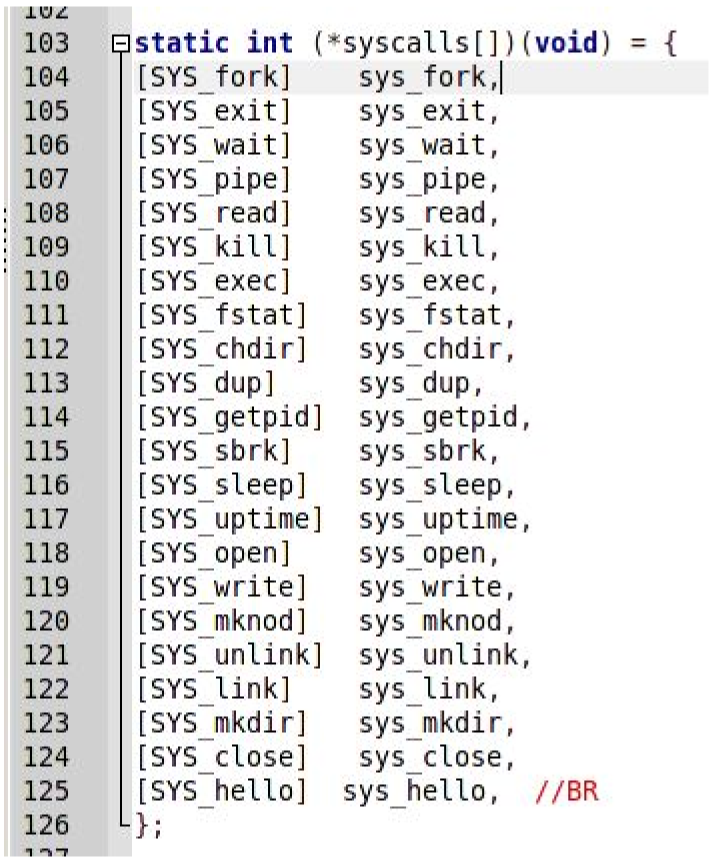
```
// System call numbers
#define SYS_fork    1
#define SYS_exit    2
#define SYS_wait    3
#define SYS_pipe    4
#define SYS_read    5
#define SYS_kill    6
#define SYS_exec    7
#define SYS_fstat   8
#define SYS_chdir   9
#define SYS_dup    10
#define SYS_getpid 11
#define SYS_sbrk   12
#define SYS_sleep  13
#define SYS_uptime 14
#define SYS_open   15
#define SYS_write  16
#define SYS_mknod  17
#define SYS_unlink 18
#define SYS_link   19
#define SYS_mkdir  20
#define SYS_close  21
#define SYS_hello  22  //BR

```

In “syscall.c”



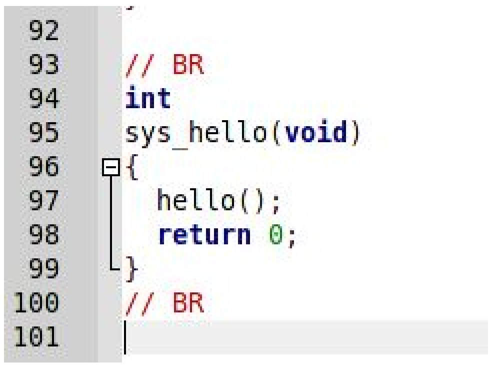
```
extern int sys_chdir(void);
extern int sys_close(void);
extern int sys_dup(void);
extern int sys_exec(void);
extern int sys_exit(void);
extern int sys_fork(void);
extern int sys_fstat(void);
extern int sys_getpid(void);
extern int sys_kill(void);
extern int sys_link(void);
extern int sys_mkdir(void);
extern int sys_mknod(void);
extern int sys_open(void);
extern int sys_pipe(void);
extern int sys_read(void);
extern int sys_sbrk(void);
extern int sys_sleep(void);
extern int sys_unlink(void);
extern int sys_wait(void);
extern int sys_write(void);
extern int sys_uptime(void);
extern int sys_hello(void);  //BR

```

And



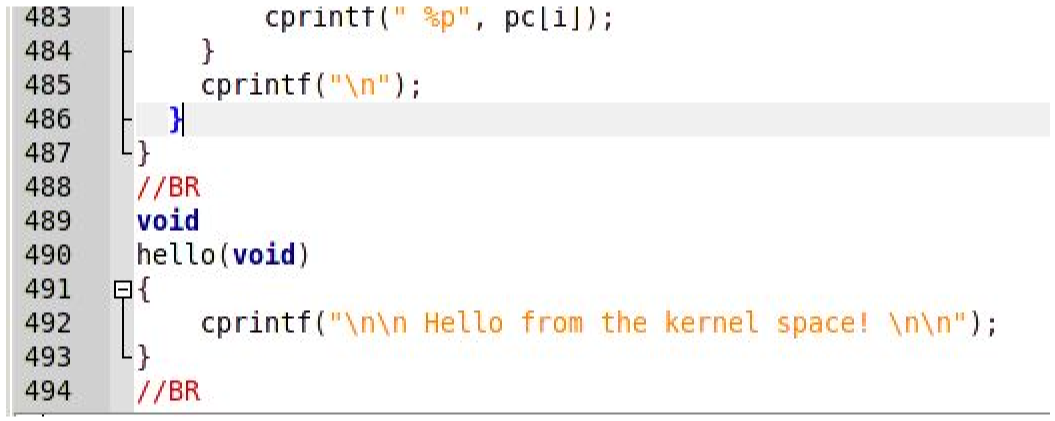
```c
static int (*syscalls[])(void) = {
[SYS_fork]    sys_fork,
[SYS_exit]    sys_exit,
[SYS_wait]    sys_wait,
[SYS_pipe]    sys_pipe,
[SYS_read]    sys_read,
[SYS_kill]    sys_kill,
[SYS_exec]    sys_exec,
[SYS_fstat]   sys_fstat,
[SYS_chdir]   sys_chdir,
[SYS_dup]     sys_dup,
[SYS_getpid]  sys_getpid,
[SYS_sbrk]    sys_sbrk,
[SYS_sleep]   sys_sleep,
[SYS_uptime]  sys_uptime,
[SYS_open]    sys_open,
[SYS_write]   sys_write,
[SYS_mknod]   sys_mknod,
[SYS_unlink]  sys_unlink,
[SYS_link]    sys_link,
[SYS_mkdir]   sys_mkdir,
[SYS_close]   sys_close,
[SYS_hello]  sys_hello,  //BR
};
```

In “sysproc.c”



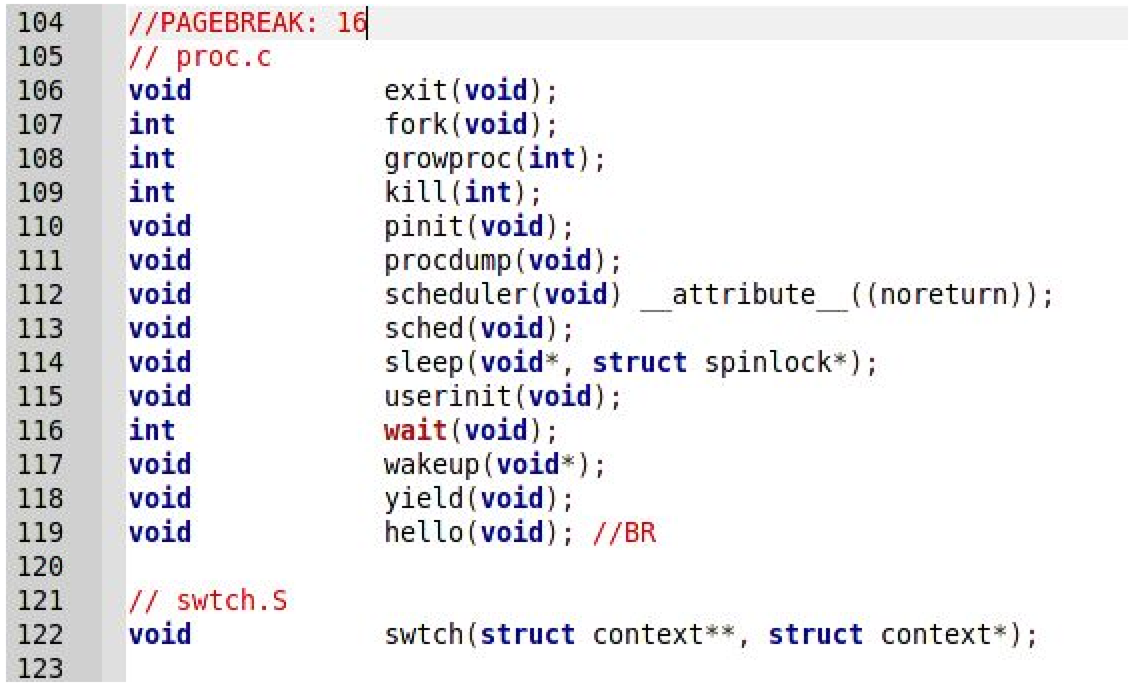
```c
// BR
int
sys_hello(void)
{
  hello();
  return 0;
}
// BR
```

In “proc.c”



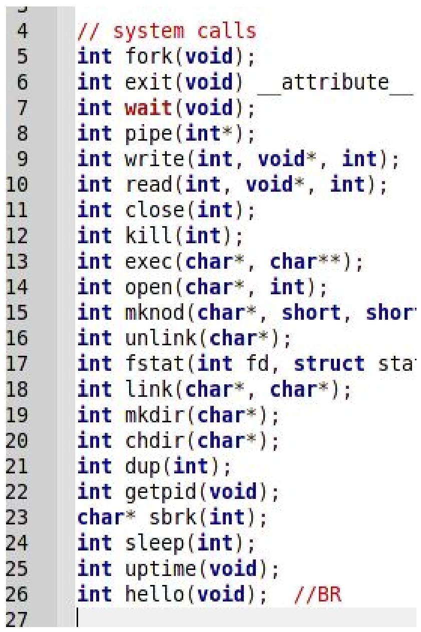
```
            cprintf(" %p", pc[i]);
        }
        cprintf("\n");
    }
}
//BR
void
hello(void)
{
    cprintf("\n\n Hello from the kernel space! \n\n");
}
//BR
```

In “defs.h”



```
//PAGEBREAK: 16
// proc.c
void            exit(void);
int             fork(void);
int             growproc(int);
int             kill(int);
void            pinit(void);
void            procdump(void);
void            scheduler(void) __attribute__((noreturn));
void            sched(void);
void            sleep(void*, struct spinlock*);
void            userinit(void);
int             wait(void);
void            wakeup(void*);
void            yield(void);
void            hello(void); //BR

// swtch.S
void            swtch(struct context**, struct context*);

```

In “user.h”

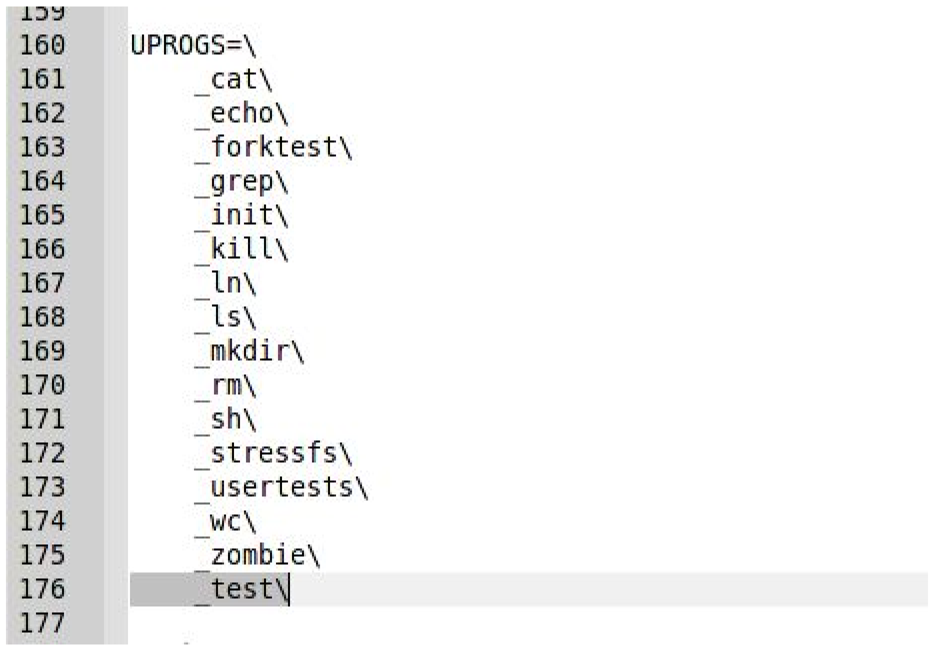


```c
// system calls
int fork(void);
int exit(void) __attribute__
int wait(void);
int pipe(int*);
int write(int, void*, int);
int read(int, void*, int);
int close(int);
int kill(int);
int exec(char*, char**);
int open(char*, int);
int mknod(char*, short, shor
int unlink(char*);
int fstat(int fd, struct sta
int link(char*, char*);
int mkdir(char*);
int chdir(char*);
int dup(int);
int getpid(void);
char* sbrk(int);
int sleep(int);
int uptime(void);
int hello(void);  //BR

```

Create “test.c” file in the home directory of “xv6-public”

```c
#include "types.h"
#include "stat.h"
#include "user.h"

int main(int argc, char *argv[])
{

  hello();
  exit();
}
```

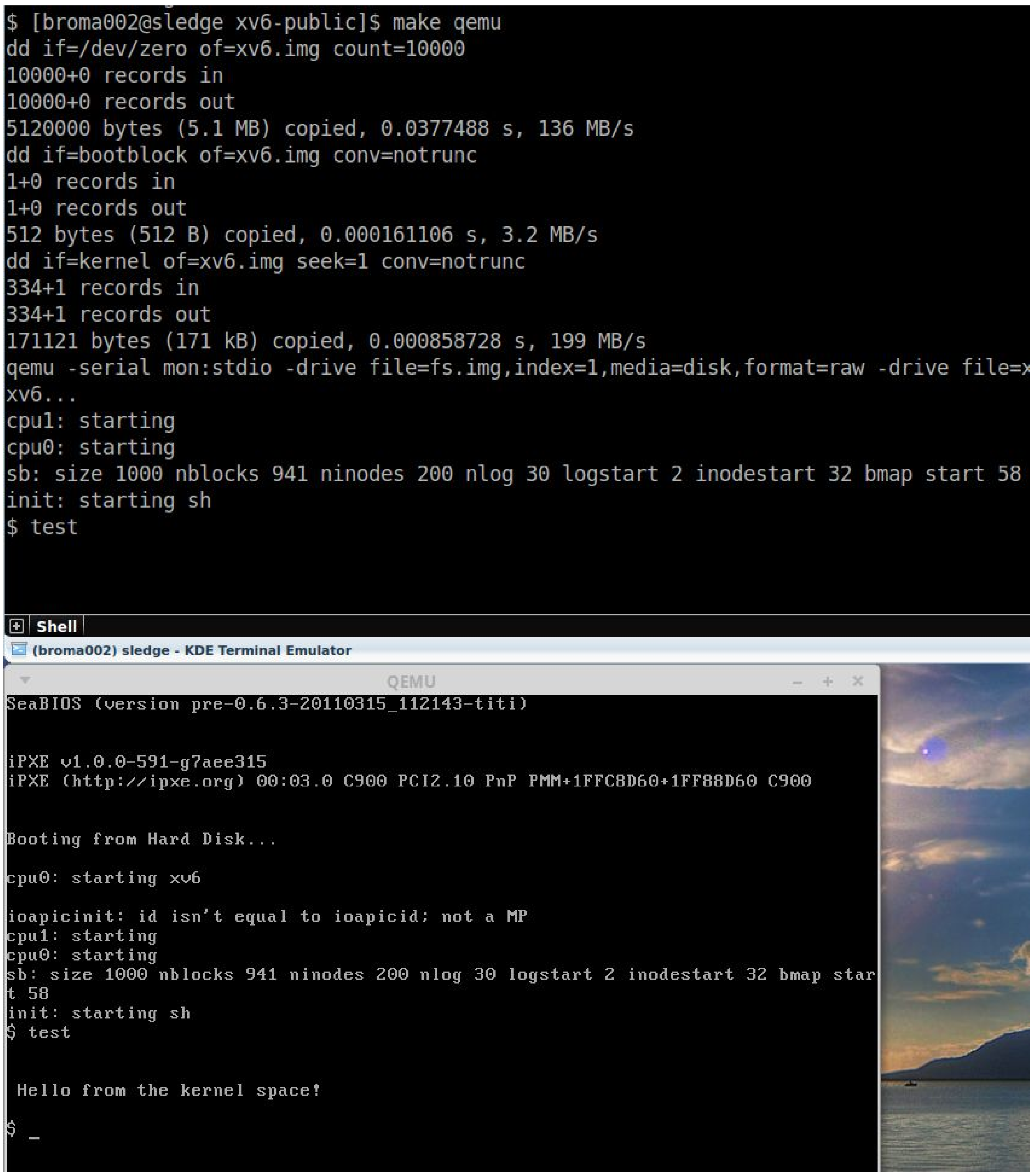
Edit “Makefile” by appending “_test\” to UPROGS



```
UPROGS=\
	_cat\
	_echo\
	_forktest\
	_grep\
	_init\
	_kill\
	_ln\
	_ls\
	_mkdir\
	_rm\
	_sh\
	_stressfs\
	_usertests\
	_wc\
	_zombie\
	_test\

```

Now type:
$ make qemu
After xv6 boots: type”
$ test

And you should see the message:



```
$ [broma002@sledge xv6-public]$ make qemu
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB) copied, 0.0377488 s, 136 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.000161106 s, 3.2 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
334+1 records in
334+1 records out
171121 bytes (171 kB) copied, 0.000858728 s, 199 MB/s
qemu -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=x
xv6...
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ test
```

```
SeaBIOS (version pre-0.6.3-20110315_112143-titi)


iPXE v1.0.0-591-g7aee315
iPXE (http://ipxe.org) 00:03.0 C900 PCI2.10 PnP PMM+1FFC8D60+1FF88D60 C900


Booting from Hard Disk...

cpu0: starting xv6

ioapicinit: id isn't equal to ioapicid; not a MP
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
$ test


 Hello from the kernel space!

$ _
```

To run Qemu with GDB, you need to open another terminal at the same xv6-public folder:

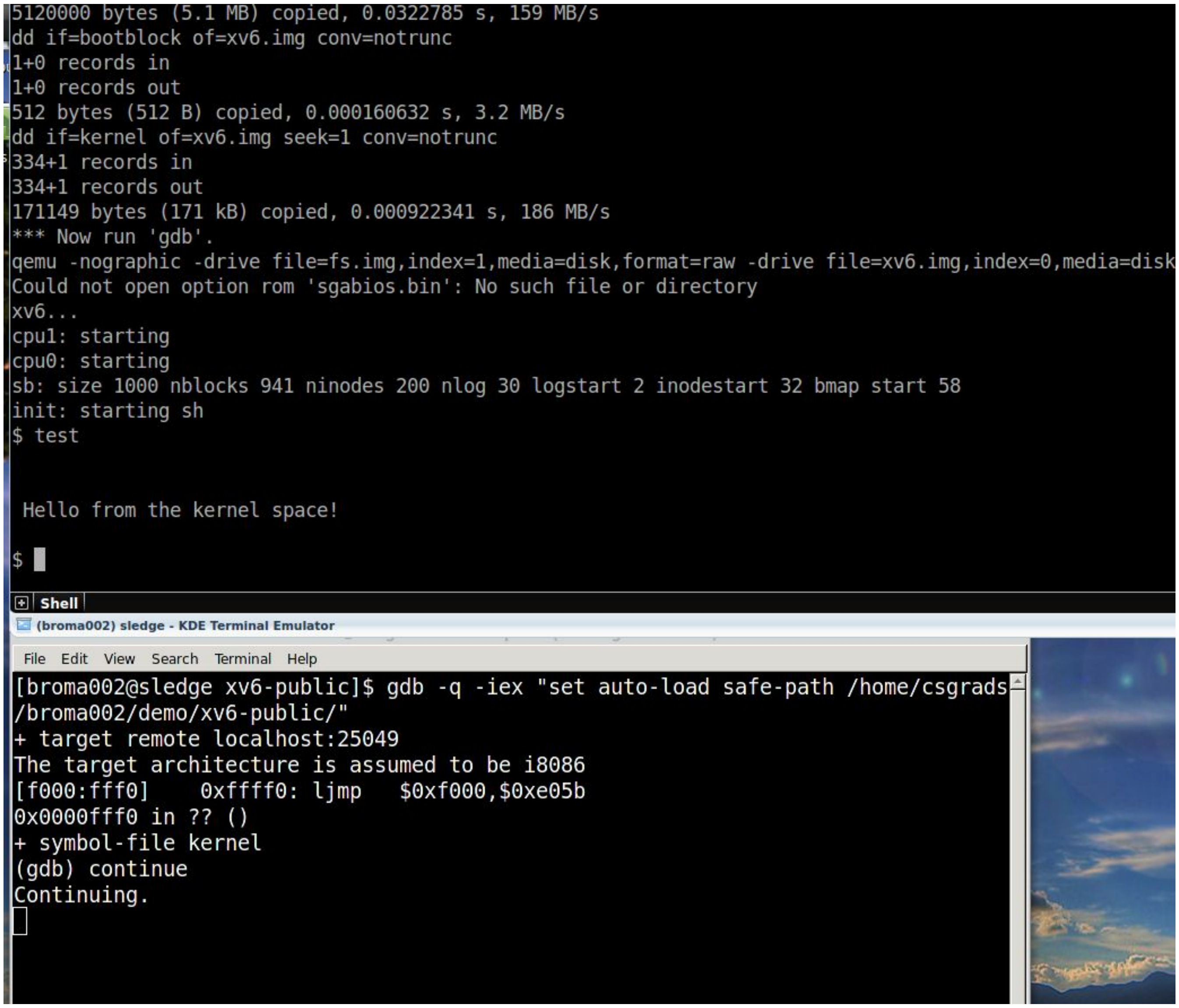
```
$ gnome-terminal&
```

In terminal 1type:

```
$ make qemu-gdb
```

In second terminal type:

```
$ gdb -q -iex "set auto-load safe-path /home/csgrads/username/xv6-public/"

+ target remote localhost:25049
The target architecture is assumed to be i8086
[f000:fff0]    0xffff0:     ljmp   $0xf000,$0xe05b
0x0000fff0 in ?? ()
+ symbol-file kernel
$ (gdb) continue
```



```
5120000 bytes (5.1 MB) copied, 0.0322785 s, 159 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.000160632 s, 3.2 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
334+1 records in
334+1 records out
171149 bytes (171 kB) copied, 0.000922341 s, 186 MB/s
*** Now run 'gdb'.
qemu -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk
Could not open option rom 'sgabios.bin': No such file or directory
xv6...
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ test


 Hello from the kernel space!

$ 
```

Shell
(broma002) sledge - KDE Terminal Emulator
File Edit View Search Terminal Help

```
[broma002@sledge xv6-public]$ gdb -q -iex "set auto-load safe-path /home/csgrads
/broma002/demo/xv6-public/"
+ target remote localhost:25049
The target architecture is assumed to be i8086
[f000:fff0]    0xffff0: ljmp   $0xf000,$0xe05b
0x0000fff0 in ?? ()
+ symbol-file kernel
(gdb) continue
Continuing.
```

Qemu uses two CPUs mode. To switch to a single core edit the Makefile:

Line 213: CPUS := 2
And replace 2 with 1.