

# How to Add a System Call in MINIX 3.1.8

By Junjie Li

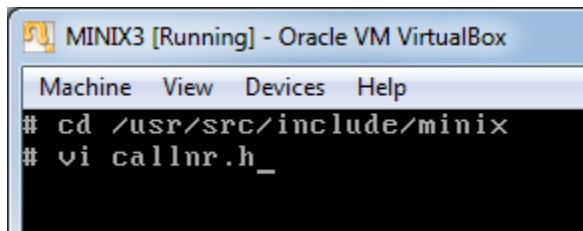
CISE Department, University of Florida

Feb. 13, 2012

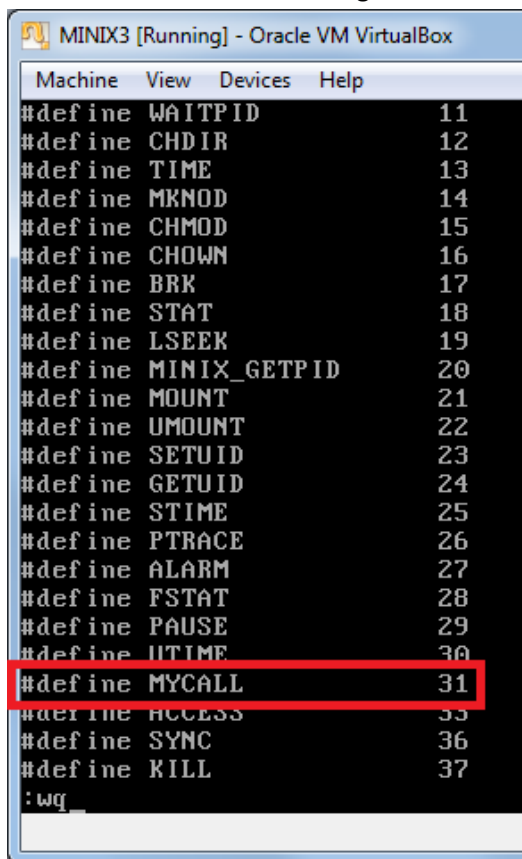
**Object:** we want to add a simple system call which prints “Hello World! This is my system call!” once called.

## Steps:

1. Edit “/usr/src/include/minix/callnr.h” and find an unused slot.

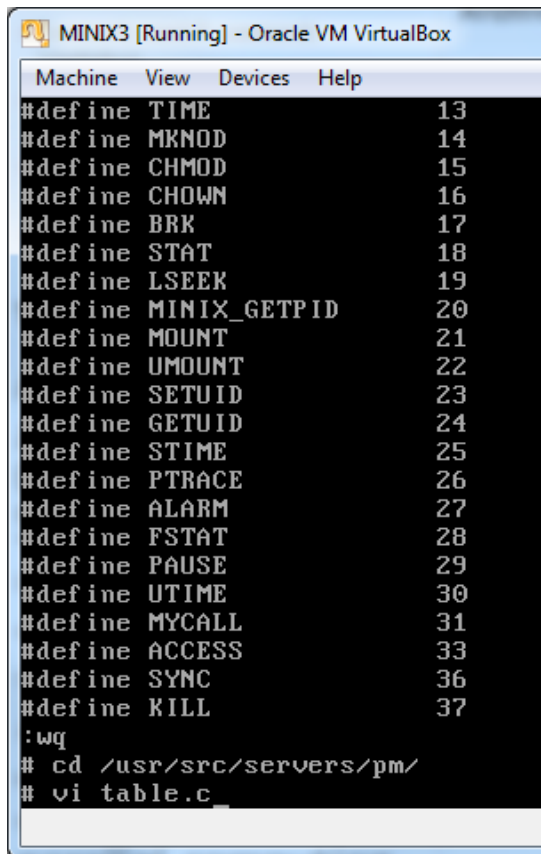


2. Here we use slot 31 and change it to “#define MYCALL 31”.



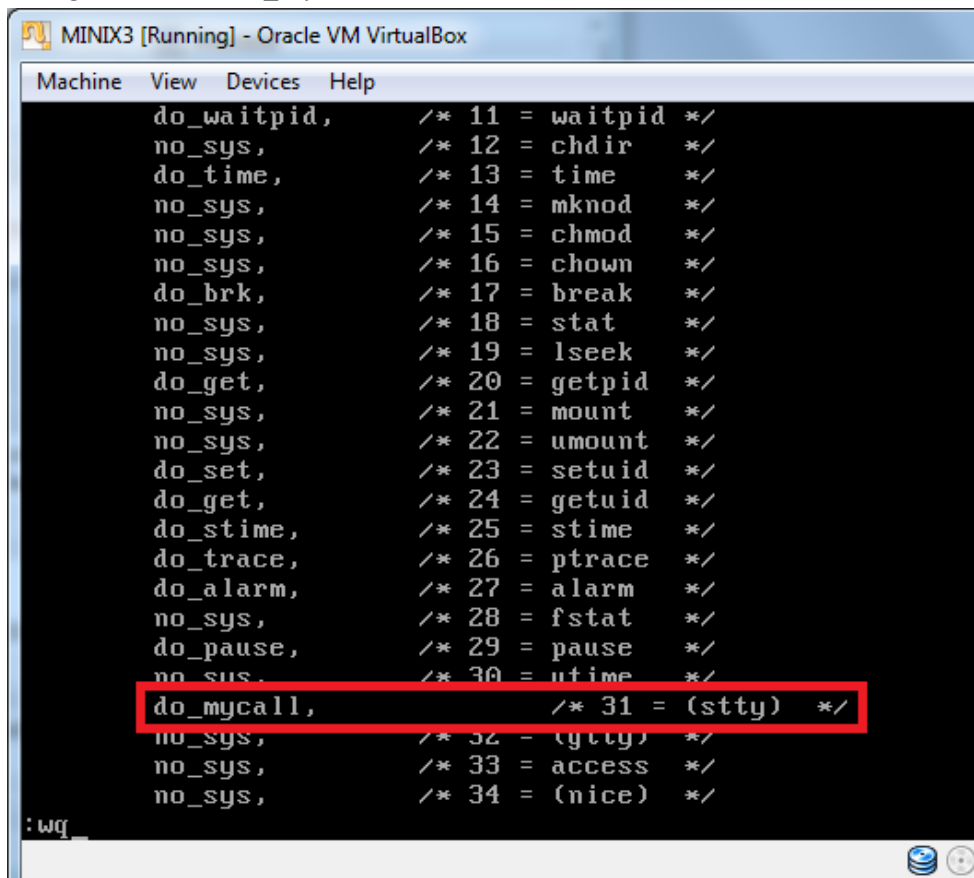
System Call	Slot
#define WAITPID	11
#define CHDIR	12
#define TIME	13
#define MKNOD	14
#define CHMOD	15
#define CHOWN	16
#define BRK	17
#define STAT	18
#define LSEEK	19
#define MINIX_GETPID	20
#define MOUNT	21
#define Umount	22
#define SETUID	23
#define GETUID	24
#define STIME	25
#define PTRACE	26
#define ALARM	27
#define FSTAT	28
#define PAUSE	29
#define UTIME	30
#define MYCALL	31
#define ACCESS	32
#define SYNC	36
#define KILL	37

3. Then edit “/usr/src/servers/pm/table.c”



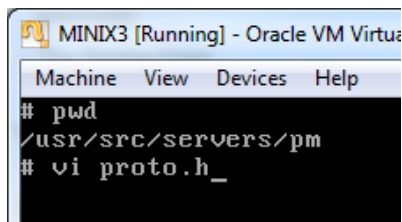
```
MINIX3 [Running] - Oracle VM VirtualBox
Machine View Devices Help
#define TIME 13
#define MKNOD 14
#define CHMOD 15
#define CHOWN 16
#define BRK 17
#define STAT 18
#define LSEEK 19
#define MINIX_GETPID 20
#define MOUNT 21
#define UMOUNT 22
#define SETUID 23
#define GETUID 24
#define STIME 25
#define PTRACE 26
#define ALARM 27
#define FSTAT 28
#define PAUSE 29
#define UTIME 30
#define MYCALL 31
#define ACCESS 33
#define SYNC 36
#define KILL 37
:wq
# cd /usr/src/servers/pm/
# vi table.c_
```

4. Change slot 31 to "do\_mycall"



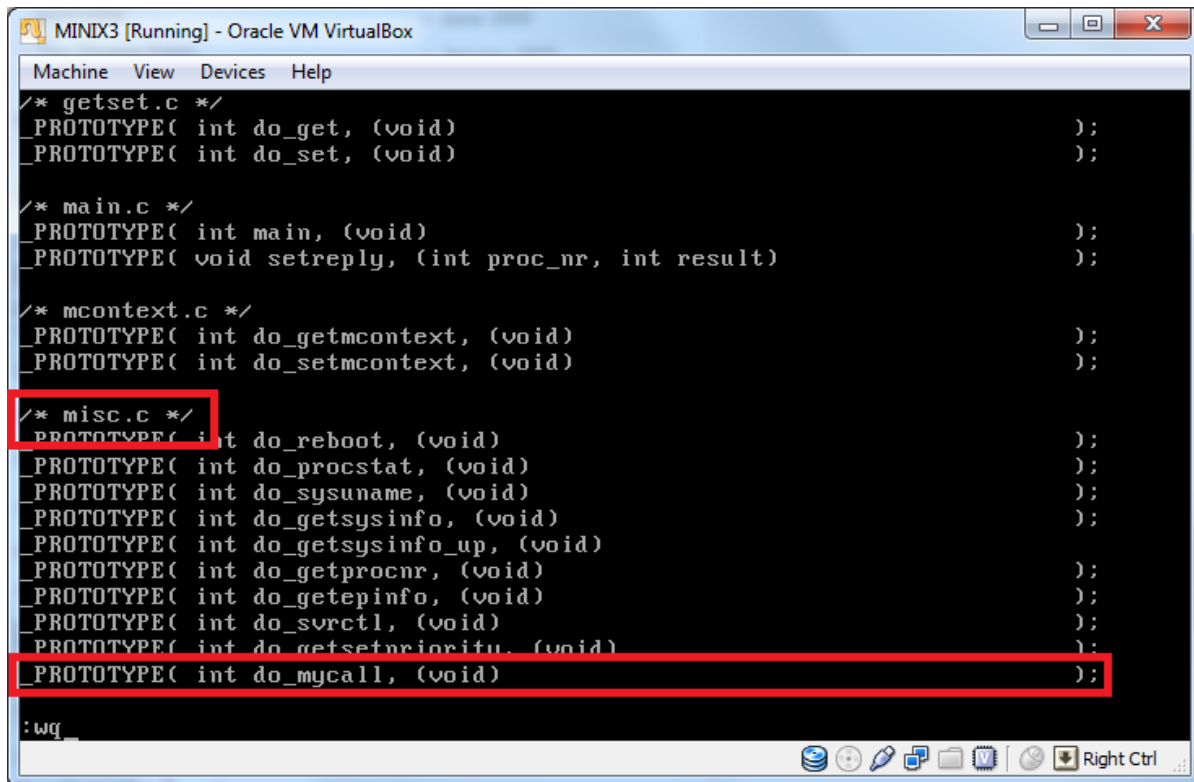
```
MINIX3 [Running] - Oracle VM VirtualBox
Machine View Devices Help
do_waitpid, /* 11 = waitpid */
no_sys, /* 12 = chdir */
do_time, /* 13 = time */
no_sys, /* 14 = mknod */
no_sys, /* 15 = chmod */
no_sys, /* 16 = chown */
do_brk, /* 17 = break */
no_sys, /* 18 = stat */
no_sys, /* 19 = lseek */
do_get, /* 20 = getpid */
no_sys, /* 21 = mount */
no_sys, /* 22 = umount */
do_set, /* 23 = setuid */
do_get, /* 24 = getuid */
do_stime, /* 25 = stime */
do_trace, /* 26 = ptrace */
do_alarm, /* 27 = alarm */
no_sys, /* 28 = fstat */
do_pause, /* 29 = pause */
no_sys, /* 30 = utime */
do_mycall, /* 31 = (stty) */
no_sys, /* 32 = (gtty) */
no_sys, /* 33 = access */
no_sys, /* 34 = (nice) */
:wq
```

5. Edit "/usr/src/servers/pm/proto.h"



```
MINIX3 [Running] - Oracle VM VirtualBox
Machine View Devices Help
# pwd
/usr/src/servers/pm
# vi proto.h_
```

6. Add the prototype in “misc.c” section



```
MINIX3 [Running] - Oracle VM VirtualBox
Machine View Devices Help
/* getset.c */
_PROTOTYPE( int do_get, (void) );
_PROTOTYPE( int do_set, (void) );

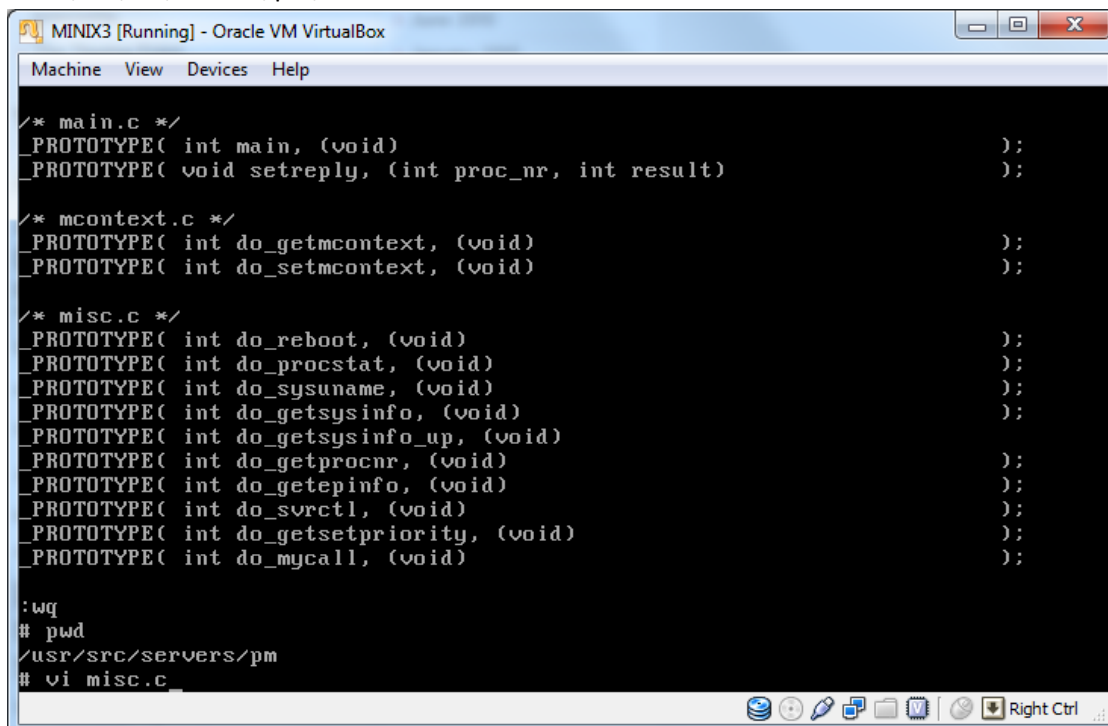
/* main.c */
_PROTOTYPE( int main, (void) );
_PROTOTYPE( void setreply, (int proc_nr, int result) );

/* mcontext.c */
_PROTOTYPE( int do_getmcontext, (void) );
_PROTOTYPE( int do_setmcontext, (void) );

/* misc.c */
_PROTOTYPE( int do_reboot, (void) );
_PROTOTYPE( int do_procstat, (void) );
_PROTOTYPE( int do_sysuname, (void) );
_PROTOTYPE( int do_getsysinfo, (void) );
_PROTOTYPE( int do_getsysinfo_up, (void) );
_PROTOTYPE( int do_getprocnr, (void) );
_PROTOTYPE( int do_getepinfo, (void) );
_PROTOTYPE( int do_svrctl, (void) );
_PROTOTYPE( int do_getsetpriority, (void) );
_PROTOTYPE( int do_mycall, (void) );

:wq_
```

7. Edit “/usr/src/servers/pm/misc.c”



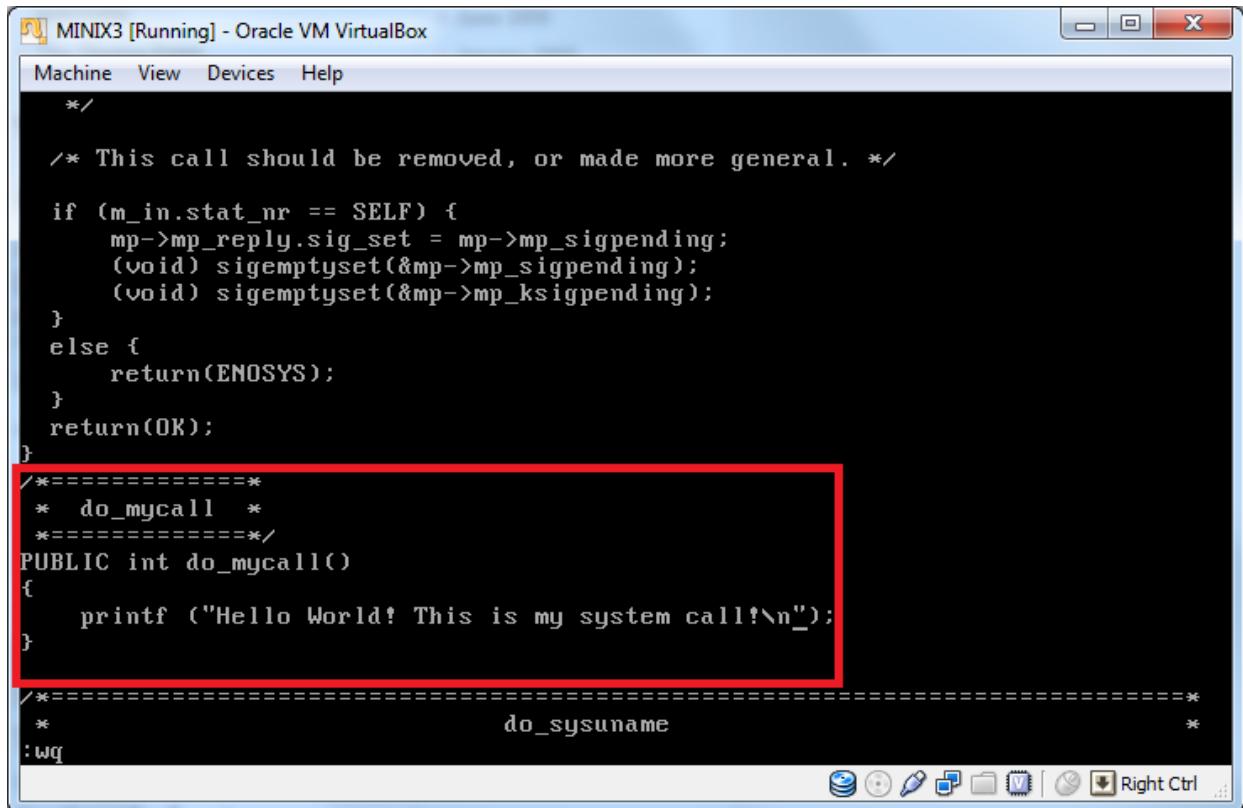
```
MINIX3 [Running] - Oracle VM VirtualBox
Machine View Devices Help
/* main.c */
_PROTOTYPE( int main, (void) );
_PROTOTYPE( void setreply, (int proc_nr, int result) );

/* mcontext.c */
_PROTOTYPE( int do_getmcontext, (void) );
_PROTOTYPE( int do_setmcontext, (void) );

/* misc.c */
_PROTOTYPE( int do_reboot, (void) );
_PROTOTYPE( int do_procstat, (void) );
_PROTOTYPE( int do_sysuname, (void) );
_PROTOTYPE( int do_getsysinfo, (void) );
_PROTOTYPE( int do_getsysinfo_up, (void) );
_PROTOTYPE( int do_getprocnr, (void) );
_PROTOTYPE( int do_getepinfo, (void) );
_PROTOTYPE( int do_svrctl, (void) );
_PROTOTYPE( int do_getsetpriority, (void) );
_PROTOTYPE( int do_mycall, (void) );

:wq
# pwd
/usr/src/servers/pm
# vi misc.c
```

8. Add the definition of the call in “misc.c”



```
MINIX3 [Running] - Oracle VM VirtualBox
Machine View Devices Help

*/

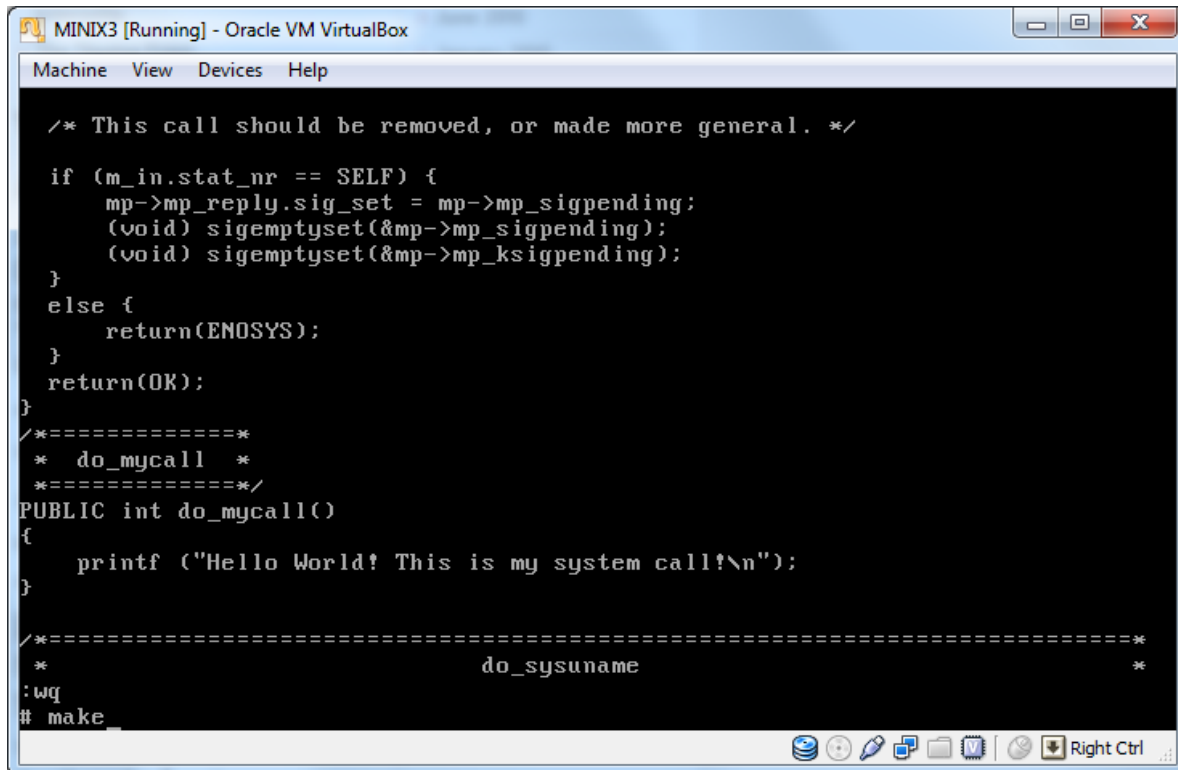
/* This call should be removed, or made more general. */

if (m_in.stat_nr == SELF) {
    mp->mp_reply.sig_set = mp->mp_sigpending;
    (void) sigemptyset(&mp->mp_sigpending);
    (void) sigemptyset(&mp->mp_ksigpending);
}
else {
    return(ENOSYS);
}
return(OK);
}

/*=====
 * do_mycall *
 *=====*/
PUBLIC int do_mycall()
{
    printf ("Hello World! This is my system call!\n");
}

/*=====
 * do_sysuname
 *=====*/
:wq
```

9. Execute “make” to check if the definition compiles.



```
MINIX3 [Running] - Oracle VM VirtualBox
Machine View Devices Help

/* This call should be removed, or made more general. */

if (m_in.stat_nr == SELF) {
    mp->mp_reply.sig_set = mp->mp_sigpending;
    (void) sigemptyset(&mp->mp_sigpending);
    (void) sigemptyset(&mp->mp_ksigpending);
}
else {
    return(ENOSYS);
}
return(OK);
}

/*=====
 * do_mycall *
 *=====*/
PUBLIC int do_mycall()
{
    printf ("Hello World! This is my system call!\n");
}

/*=====
 * do_sysuname
 *=====*/
:wq
# make
```

10. Create a file named “mycalllib.h” under “/usr/include”.

```
Machine View Devices Help
create pm/signal.d
create pm/table.d
create pm/time.d
create pm/trace.d
create pm/utility.d
create pm/.depend
compile pm/main.o
compile pm/forkexit.o
compile pm/break.o
compile pm/exec.o
compile pm/time.o
compile pm/alarm.o
compile pm/signal.o
compile pm/utility.o
compile pm/table.o
compile pm/trace.o
compile pm/getset.o
compile pm/misc.o
compile pm/profile.o
compile pm/dma.o
compile pm/mcontext.o
compile pm/schedule.o
link pm/pm
# cd /usr/include/
# vi mycalllib.h
```

11. The content of the file “mycalllib.h”

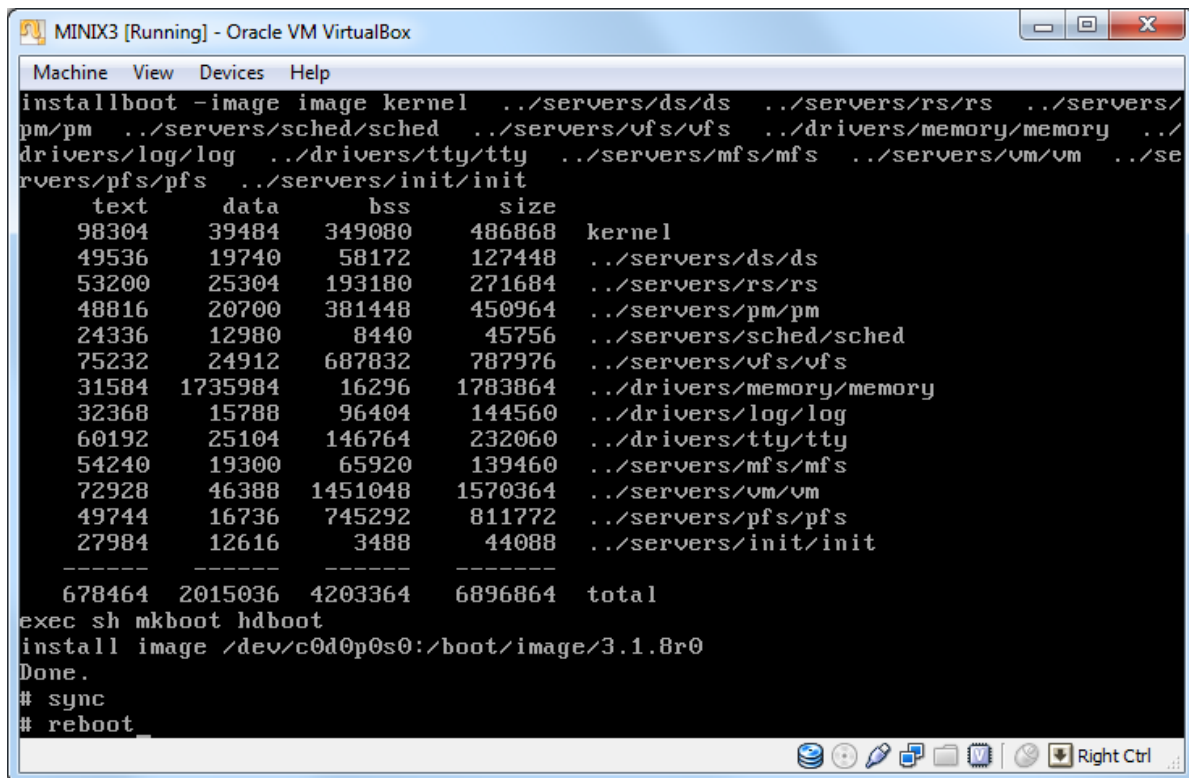
```
Machine View Devices Help
#include <lib.h>
#include <unistd.h>

PUBLIC int mycall ()
{
    message m;
    return ( _syscall(PM_PROC_NR, MYCALL, &m) );
}
```

12. Compile the new system.

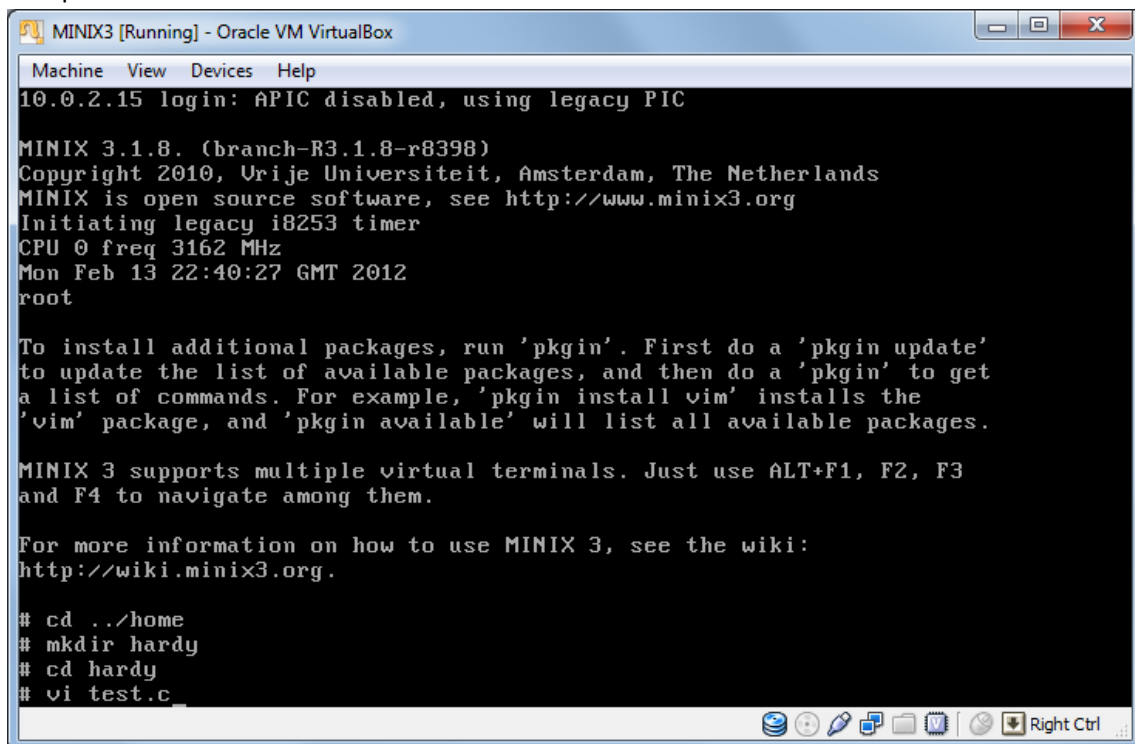
```
# cd /usr/src/tools
# make hdbboot
```

13. “sync” and “reboot”



```
MINIX3 [Running] - Oracle VM VirtualBox
Machine View Devices Help
installboot -image image kernel ../servers/ds/ds ../servers/rs/rs ../servers/
pm/pm ../servers/sched/sched ../servers/vfs/vfs ../drivers/memory/memory ../
drivers/log/log ../drivers/tty/tty ../servers/mfs/mfs ../servers/vm/vm ../se
rvers/pfs/pfs ../servers/init/init
text      data      bss      size
98304     39484     349080   486868   kernel
49536     19740     58172    127448   ../servers/ds/ds
53200     25304     193180   271684   ../servers/rs/rs
48816     20700     381448   450964   ../servers/pm/pm
24336     12980     8440     45756    ../servers/sched/sched
75232     24912     687832   787976   ../servers/vfs/vfs
31584     1735984   16296    1783864  ../drivers/memory/memory
32368     15788     96404    144560   ../drivers/log/log
60192     25104     146764   232060   ../drivers/tty/tty
54240     19300     65920    139460   ../servers/mfs/mfs
72928     46388     1451048  1570364  ../servers/vm/vm
49744     16736     745292   811772   ../servers/pfs/pfs
27984     12616     3488     44088    ../servers/init/init
-----
678464    2015036   4203364   6896864   total
exec sh mkboot hdboot
install image /dev/c0d0p0s0:/boot/image/3.1.8r0
Done.
# sync
# reboot
```

14. Compose a test file.



```
MINIX3 [Running] - Oracle VM VirtualBox
Machine View Devices Help
10.0.2.15 login: APIC disabled, using legacy PIC

MINIX 3.1.8. (branch-R3.1.8-r8398)
Copyright 2010, Urije Universiteit, Amsterdam, The Netherlands
MINIX is open source software, see http://www.minix3.org
Initiating legacy i8253 timer
CPU 0 freq 3162 MHz
Mon Feb 13 22:40:27 GMT 2012
root

To install additional packages, run 'pkgin'. First do a 'pkgin update'
to update the list of available packages, and then do a 'pkgin' to get
a list of commands. For example, 'pkgin install vim' installs the
'vim' package, and 'pkgin available' will list all available packages.

MINIX 3 supports multiple virtual terminals. Just use ALT+F1, F2, F3
and F4 to navigate among them.

For more information on how to use MINIX 3, see the wiki:
http://wiki.minix3.org.

# cd ../home
# mkdir hardy
# cd hardy
# vi test.c
```

15. The test file looks like this:



17. Compile and run. You should all be set!

The screenshot shows a window titled "MINIX3 [Running] - Oracle VM VirtualBox". Inside the window is a terminal window with a black background and white text. The terminal displays C code at the top: 

```
{  
    mycall ();  
    return 0;  
}
```

 Below this are several tilde (~) characters representing newlines. Further down, the following commands and output are shown:

```
# gcc test.c -o test.out  
# ./test.out  
Hello World! This is my system call!  
#
```

 At the bottom right of the terminal window, there is a status bar with icons for disk, CD-ROM, network, printer, folder, and a small icon with the letter 'V'. To the right of these icons, it says "Right Ctrl". The main window has a menu bar with "Machine", "View", "Devices", and "Help". The title bar includes standard Windows-style minimize, maximize, and close buttons.