

Introduction to OS161



Politecnico
di Torino

Department of Control and
Computer Engineering



OS161 kit setup

Building the kernel

Running the kernel

System Programming - Sarah Azimi

CAD & Reliability Group

DAUIN- Politecnico di Torino

Outline

- OS161 setup
- Understanding OS161
- Building OS161
- Running OS161

Os161

- OS 161 is a simplified operating system, (written in C) a simplified unix BSD-like OS.
- There are two supported branches of OS/161:
 - 1.x branch, first launched in 2001, provides a uniprocessor kernel programming environment.
 - 2.x branch, which debuted in 2009 and was finally fully released in 2015, moves into the multicore era by adding multiprocessor support and other more modern attributes.
- Includes both a kernel of conventional ("macrokernel") design and a simple userland, including a variety of test programs.

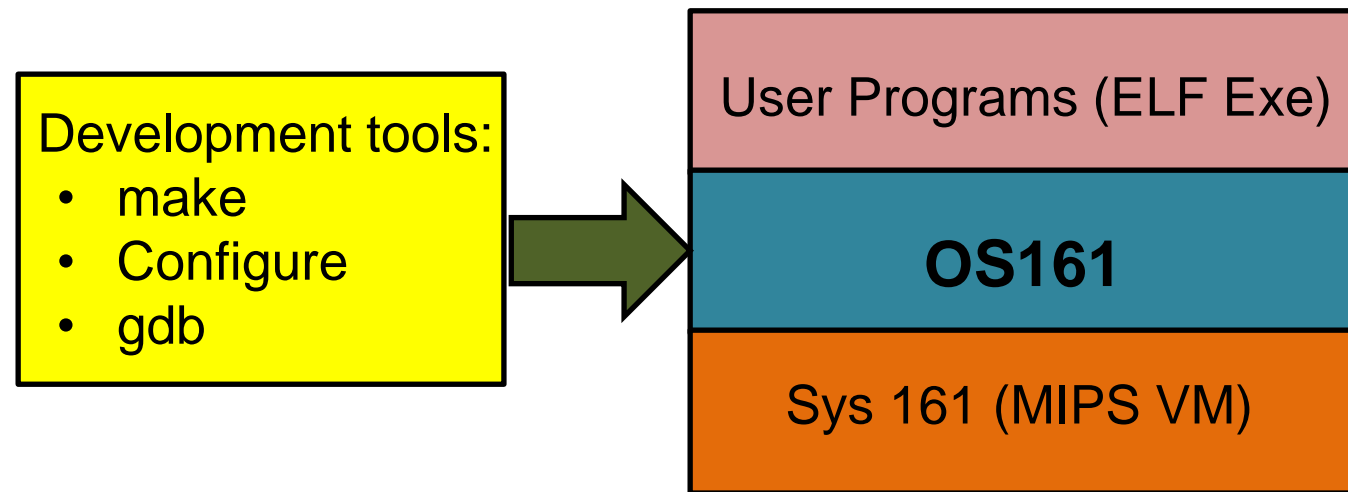
OS161 framework

OS161 includes

- The sources of the operating system (kernel), to be used for
 - Code browsing
 - Designing, implementing new/missing features
 - Running and debugging
- **A toolchain for**
 - Cross compiling (OS161 kernel for a MIPS processor)
 - Running the kernel on top of a machine simulator called sys161
 - Other tasks...

About System/161

- System/161 is a machine simulator that provides a simplified but still realistic environment for OS hacking.
- It is a 32-bit MIPS system supporting up to 32 processors.
- It was designed to support OS/161
- System/161 supports fully transparent debugging, via remote gdb into the simulator.



Os161 Support

- The base OS161 system **provides** low-level trap/interrupt, device drivers, in-kernel threads, a baseline scheduler, a minimal virtual memory system, a simple file system.
- Other things (not included) **have to be implemented**:
 - Locks
 - System calls
 - **Virtual memory** - The "dumbvm" shipped with OS161 is good enough for bootstrapping and doing the early assignments. It never reuses memory and cannot support large processes or malloc.
 - **File system**
- Many other things can be added to OS161

Understanding Os161 (First Lab)

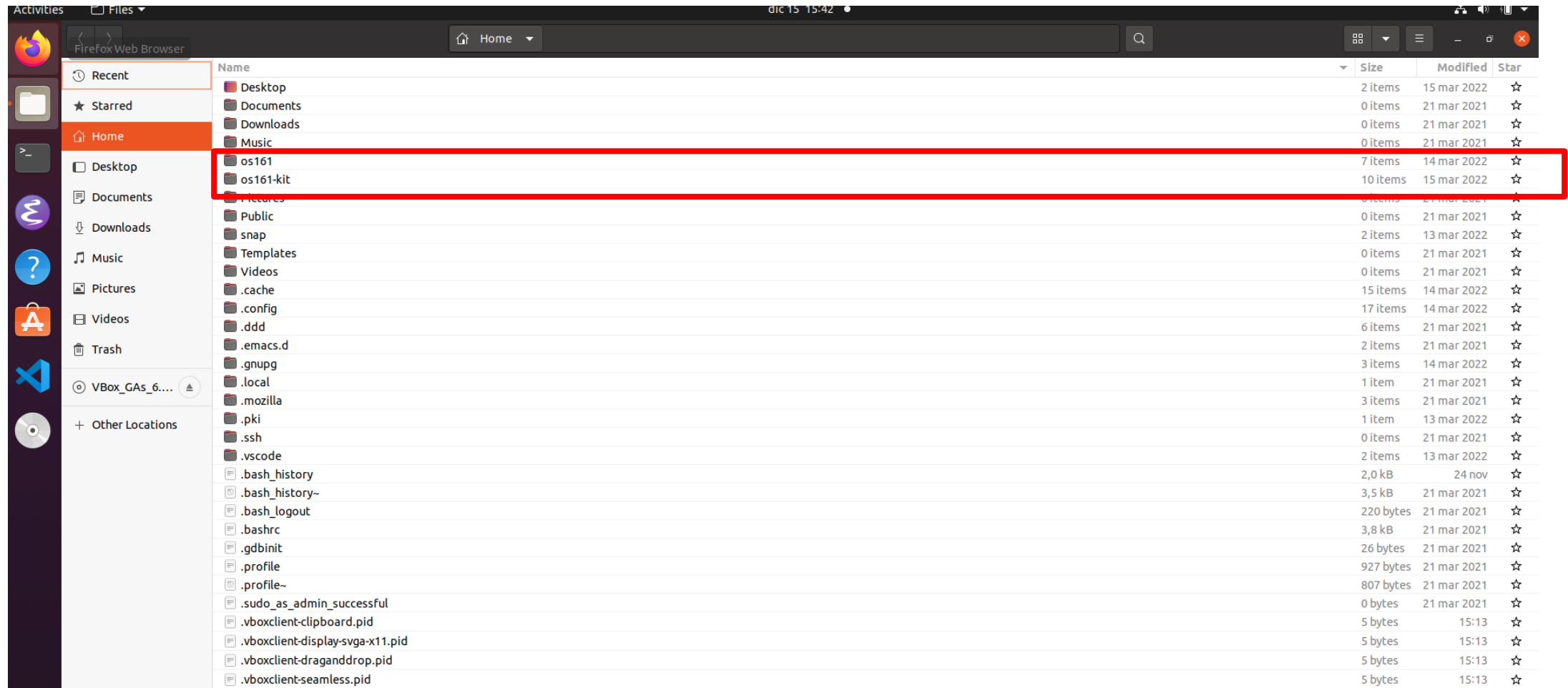
- Set up OS161 development environment.
- Understand the source code structure of OS161.
- Navigate the OS/161 sources to determine where and how things are done.
- Be able to modify, build (configure, bmake) and run OS/161 kernel.
- Use GDB.

Working in Os161

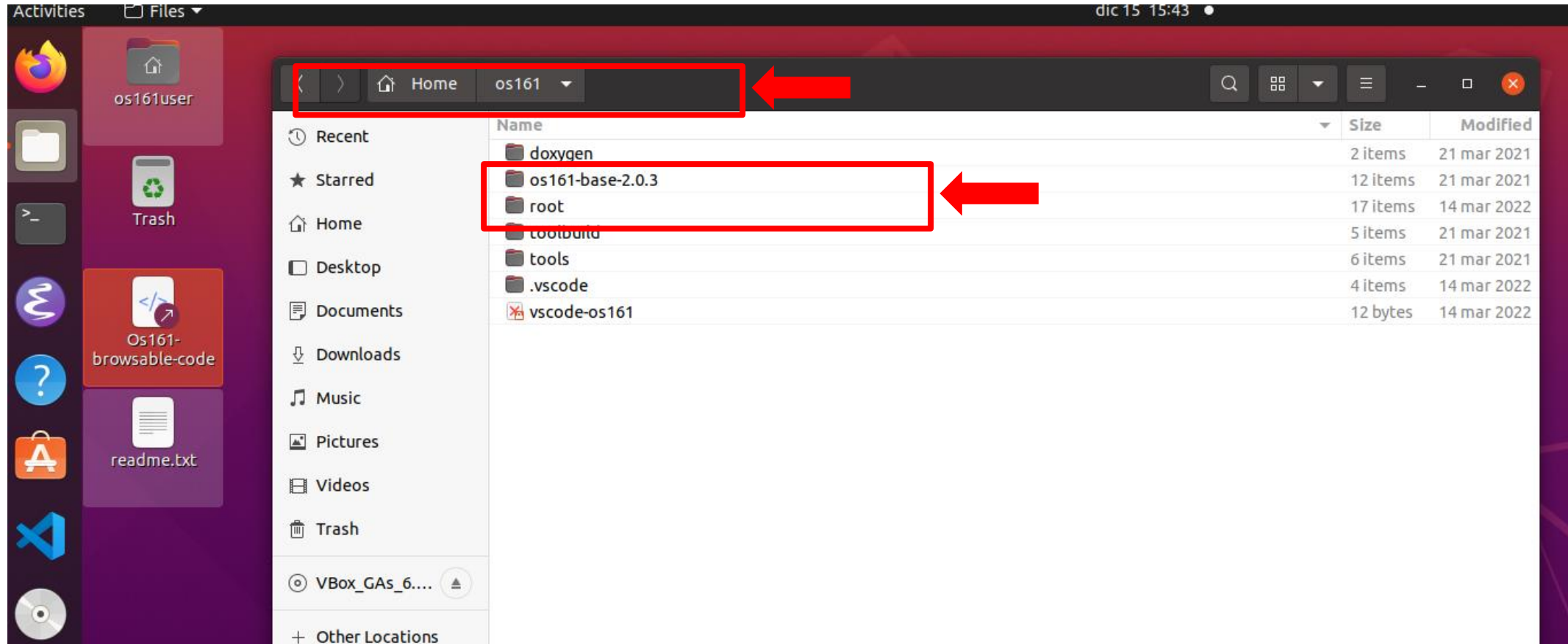
Directory tree (PdS Ubuntu 14.04 virtual machine):

- /home/pds: pds user directory
- os161_doc: documentation (with browsable code)
- pds-os161/root (full path: /home/pds/pds-os161/root): run/execution
 - pds-os161/root/testscripts: user program execs (from userland) to be called within os161 test menu.
- os161 (full path: /home/pds/os161): tools and os161 source/build
 - os161/tools: tools for compilation, make(build), debug (eg. mips-harvard.os161- gcc)
 - os161/os161-base-2.0.2: building kernel and user programs
 - os161/os161-base-2.0.2/userland: user source programs (e.g. test)
 - **os161/os161-base-2.0.2/kern: kernel source**
 - os161/os161-base-2.0.2/kern/conf: kernel configuration
 - os161/os161-base-2.0.2/kern/compile: kernel compilation/build

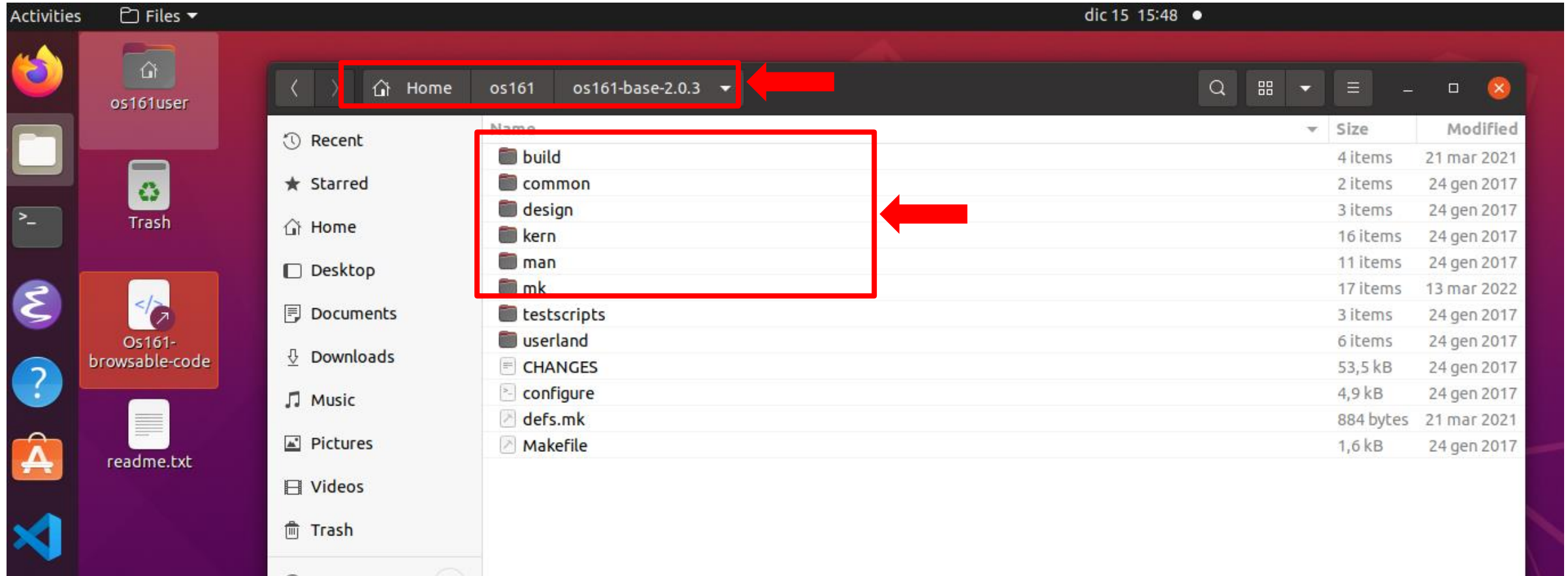
Working in Os161



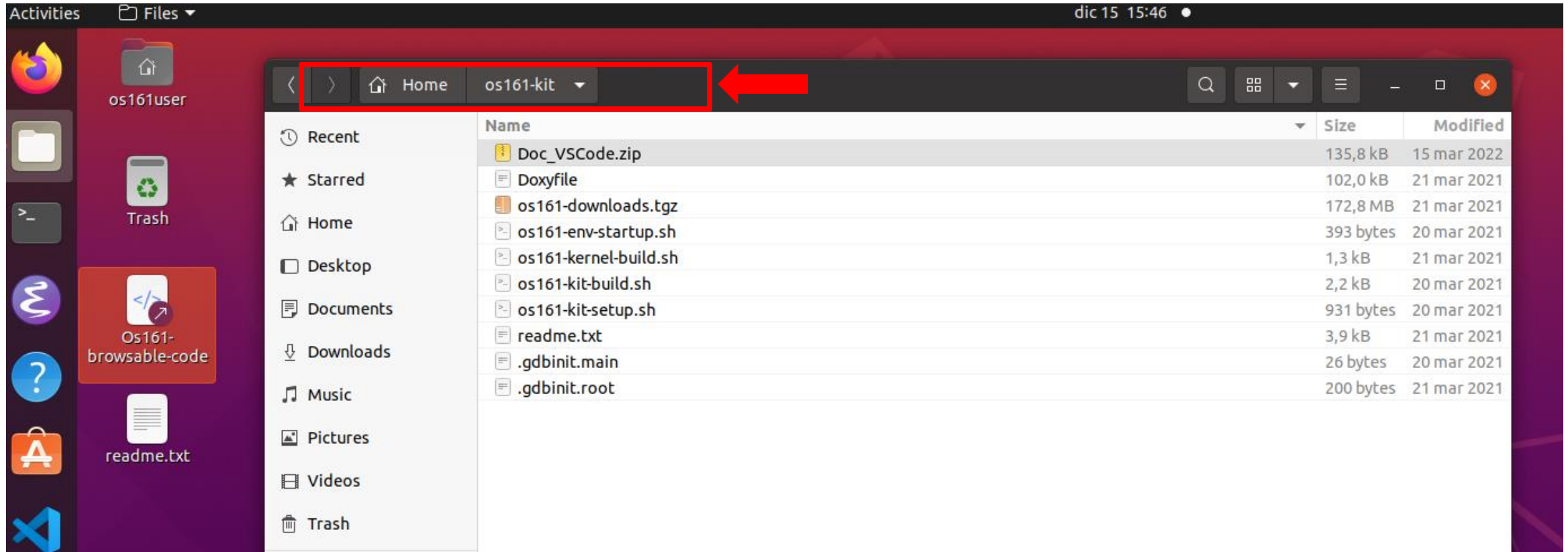
Working in Os161



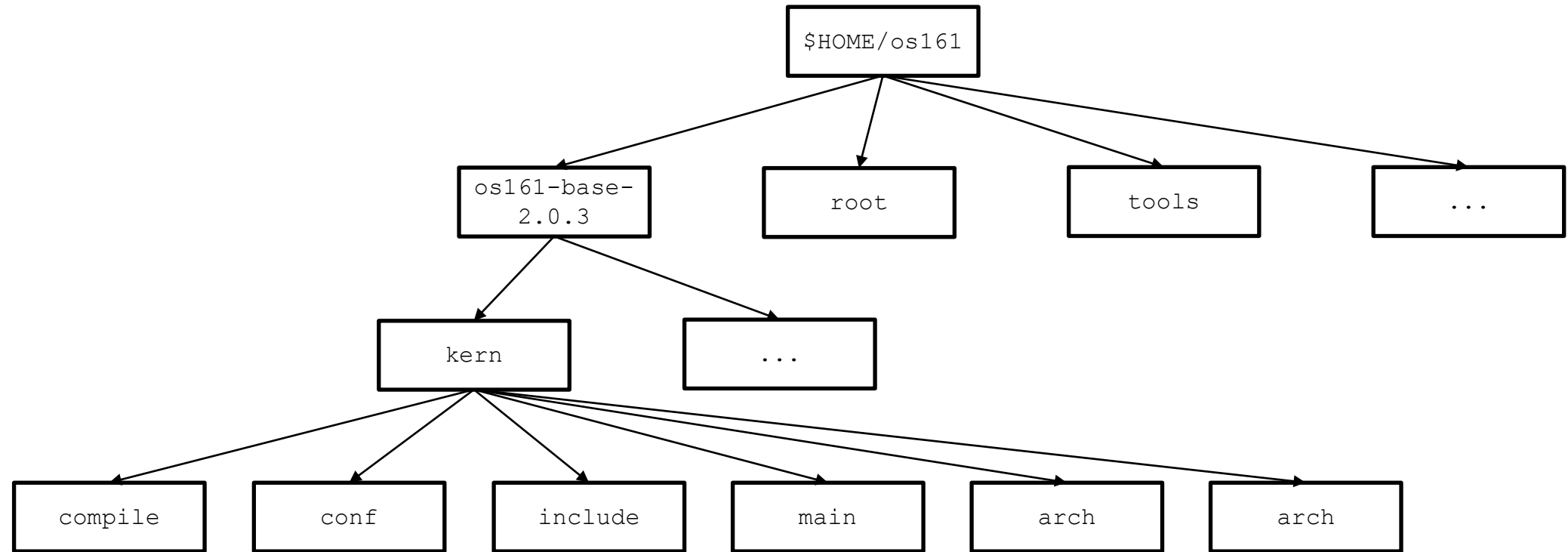
Working in Os161



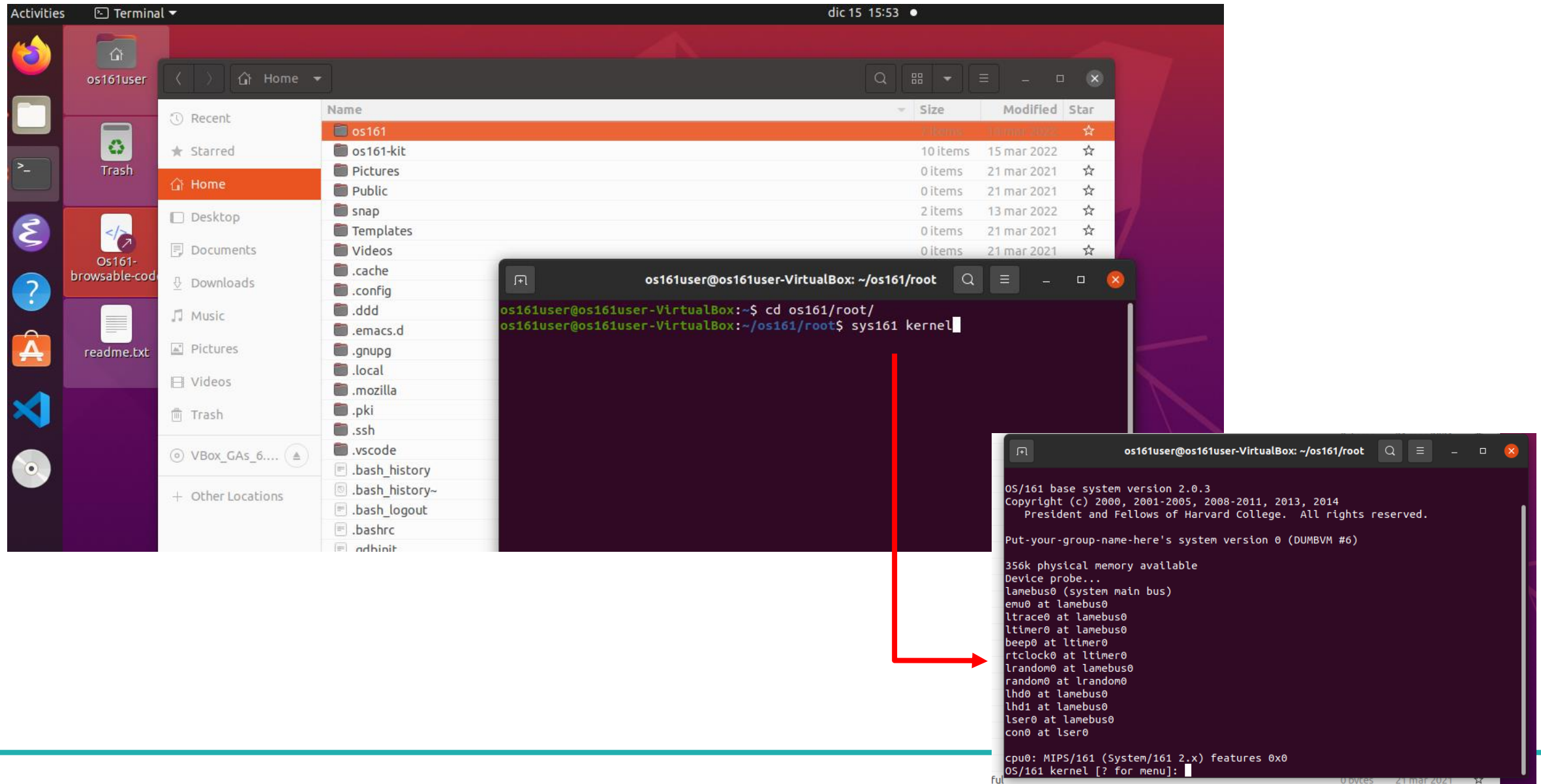
Working in Os161



OS161 Directory Tree



Making (building) OS161: Esequire



The screenshot shows a Linux desktop environment with a file manager and two terminal windows. The file manager window displays the contents of the `~/os161` directory, with the `os161` directory selected. The terminal window shows the command `sys161 kernel` being executed. A second terminal window shows the output of the command, including system version information and hardware details.

File Manager Window:

Name	Size	Modified	Star
os161	0 items	15 mar 2022	☆
os161-kit	10 items	15 mar 2022	☆
Pictures	0 items	21 mar 2021	☆
Public	0 items	21 mar 2021	☆
snap	2 items	13 mar 2022	☆
Templates	0 items	21 mar 2021	☆
Videos	0 items	21 mar 2021	☆
.cache			
.config			
.ddd			
.emacs.d			
.gnupg			
.local			
.mozilla			
.pki			
.ssh			
.vscode			
.bash_history			
.bash_history~			
.bash_logout			
.bashrc			
.rdhinit			

Terminal Window 1:

```
os161user@os161user-VirtualBox: ~/os161/root
os161user@os161user-VirtualBox:~$ cd os161/root/
os161user@os161user-VirtualBox:~/os161/root$ sys161 kernel
```

Terminal Window 2:

```
os161user@os161user-VirtualBox: ~/os161/root

OS/161 base system version 2.0.3
Copyright (c) 2000, 2001-2005, 2008-2011, 2013, 2014
  President and Fellows of Harvard College. All rights reserved.

Put-your-group-name-here's system version 0 (DUMBVM #6)

356k physical memory available
Device probe...
lamebus0 (system main bus)
emu0 at lamebus0
ltrance0 at lamebus0
ltimer0 at lamebus0
beep0 at ltimer0
rtclock0 at ltimer0
lrando0 at lamebus0
random0 at lrando0
lhd0 at lamebus0
lhd1 at lamebus0
lser0 at lamebus0
con0 at lser0

cpu0: MIPS/161 (System/161 2.x) features 0x0
OS/161 kernel [? for menu]:
```

Making (building) OS161: Esequire Debugger

- Debug mode: Opening two terminals – Option 1
 - Sys161 -w kernel
 - Mips-hardvard-os161-gdb -tui kernel

```
os161user@os161user-VirtualBox: ~/os161/root
os161user@os161user-VirtualBox:~/os161/root$ sys161 -w kernel
sys161: System/161 release 2.0.8 compiled Mar 21 2021 16:01:30
sys161: Waiting for debugger connection...

os161user@os161user-VirtualBox: ~/os161/root
os161user@os161user-VirtualBox:~/os161/root$ mips-hardvard-os161-gdb -tui kernel
os161user@os161user-VirtualBox:~/os161/root$
```


Making (building) OS161: Eseguire Debugger

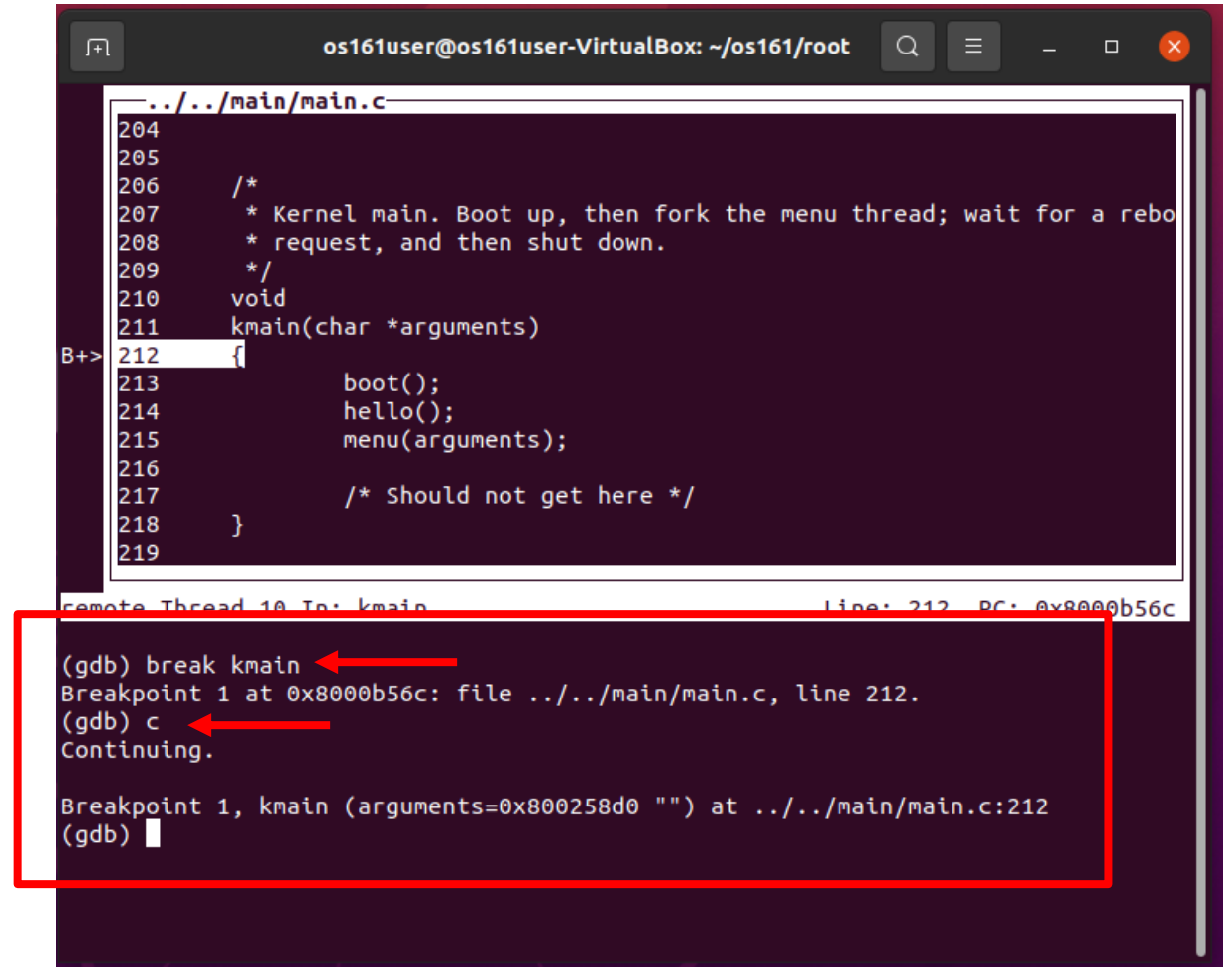
- Debug mode: Opening two terminals – Option 1
 - Sys161 -w kernel
 - Mips-hardvard-os161-gdb -tui kernel

```
os161user@os161user-VirtualBox: ~/os161/root
os161user@os161user-VirtualBox:~/os161/root$ sys161 -w kernel
sys161: System/161 release 2.0.8, compiled Mar 21 2021 16:01:33
sys161: Waiting for debugger connection...
sys161: New debugger connection
[ No Source Available ]
remote Thread 10 In: __start Line: 54 PC: 0x8001ca90
For help, type "help".
---Type <return> to continue, or q <return> to quit---return
Type "apropos word" to search for commands related to "word"...
Reading symbols from kernel...done.
Warning: /home/os161user/os161/root/./os161-base-2.0.3/kern/compile/DUMBVM: No
such file or directory.
__start () at ../../arch/sys161/main/start.S:54
(gdb)

os161user@os161user-VirtualBox: ~/os161/root
os161user@os161user-VirtualBox:~/os161/root$ sys161 -w kernel
os161user@os161user-VirtualBox: ~/os161/root
os161user@os161user-VirtualBox:~/os161/root$ mips-hardvard-os161-gdb -tui kernel
os161user@os161user-VirtualBox:~/os161/root$
```


Making (building) OS161: Esequire Debugger

- Debug mode: Opening two terminals – Option 1
 - *break kmain*
 - *C //continue*
 - Not very friendly



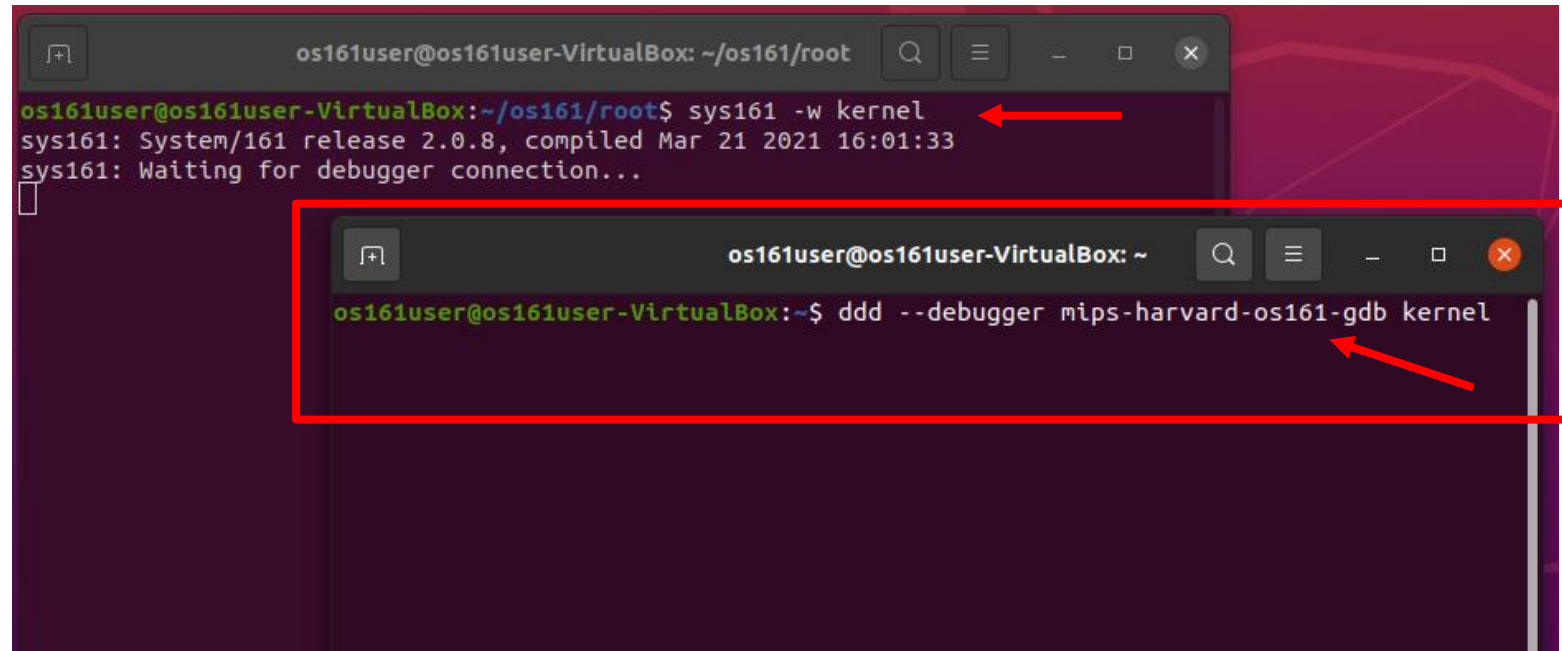
The screenshot shows a terminal window titled "os161user@os161user-VirtualBox: ~/os161/root". The main window displays the source code for `main.c` with line numbers 204 to 219. The code includes a comment about kernel main, a `kmain` function signature, and the start of the function body. A GDB prompt `B+>` is visible on line 212. Below the code window, a GDB session is shown in a separate terminal window, with a red box highlighting the commands `(gdb) break kmain` and `(gdb) c`. The output shows a breakpoint set at line 212 and the program continuing.

```
os161user@os161user-VirtualBox: ~/os161/root
../../main/main.c
204
205
206 /*
207  * Kernel main. Boot up, then fork the menu thread; wait for a rebo
208  * request, and then shut down.
209  */
210 void
211 kmain(char *arguments)
212 {
213     boot();
214     hello();
215     menu(arguments);
216
217     /* Should not get here */
218 }
219
(gdb) break kmain
Breakpoint 1 at 0x8000b56c: file ../../main/main.c, line 212.
(gdb) c
Continuing.

Breakpoint 1, kmain (arguments=0x800258d0 "") at ../../main/main.c:212
(gdb)
```

Making (building) OS161: Eseguire Debugger

- Debug mode: Opening two terminals – Option 2
 - *sys11 -w kernel*
 - *ddd --debugger mips-Harvard-os161-gdb kernel*

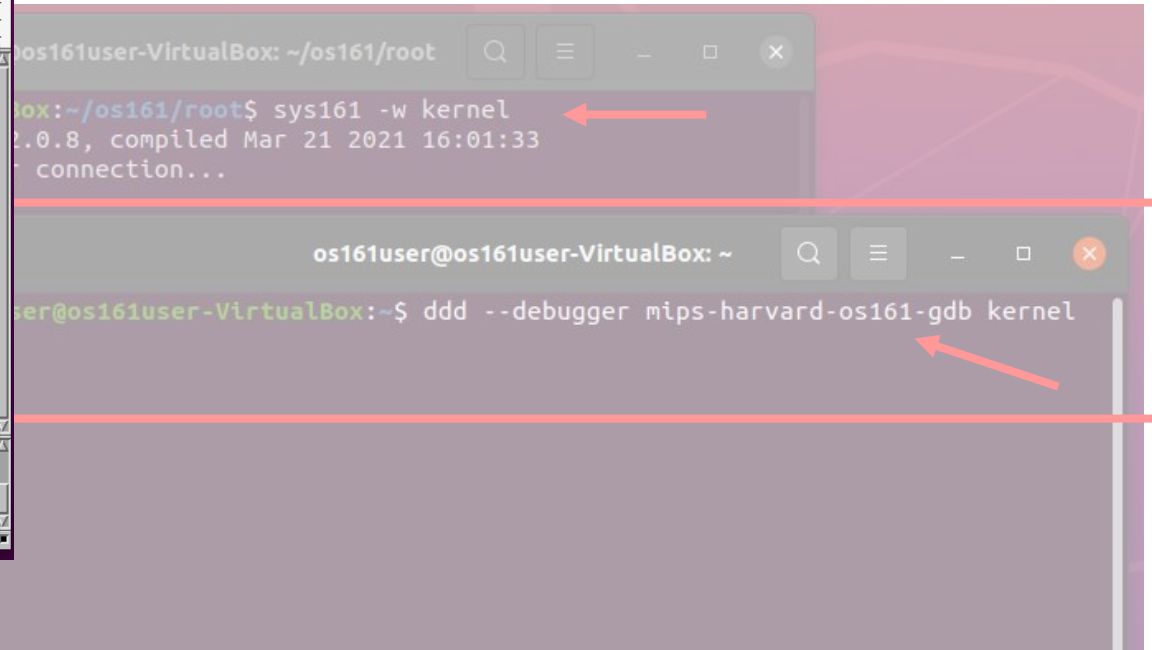
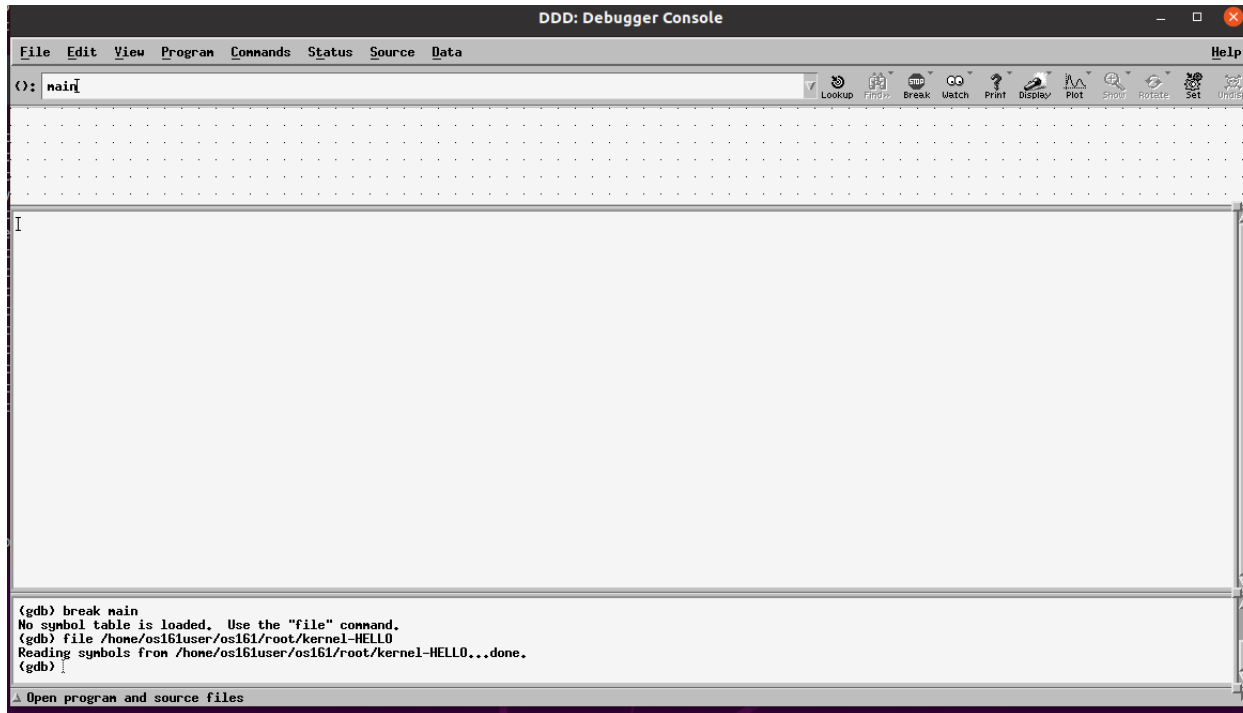


```
os161user@os161user-VirtualBox: ~/os161/root
os161user@os161user-VirtualBox:~/os161/root$ sys161 -w kernel
sys161: System/161 release 2.0.8, compiled Mar 21 2021 16:01:33
sys161: Waiting for debugger connection...

os161user@os161user-VirtualBox: ~
os161user@os161user-VirtualBox:~$ ddd --debugger mips-harvard-os161-gdb kernel
```

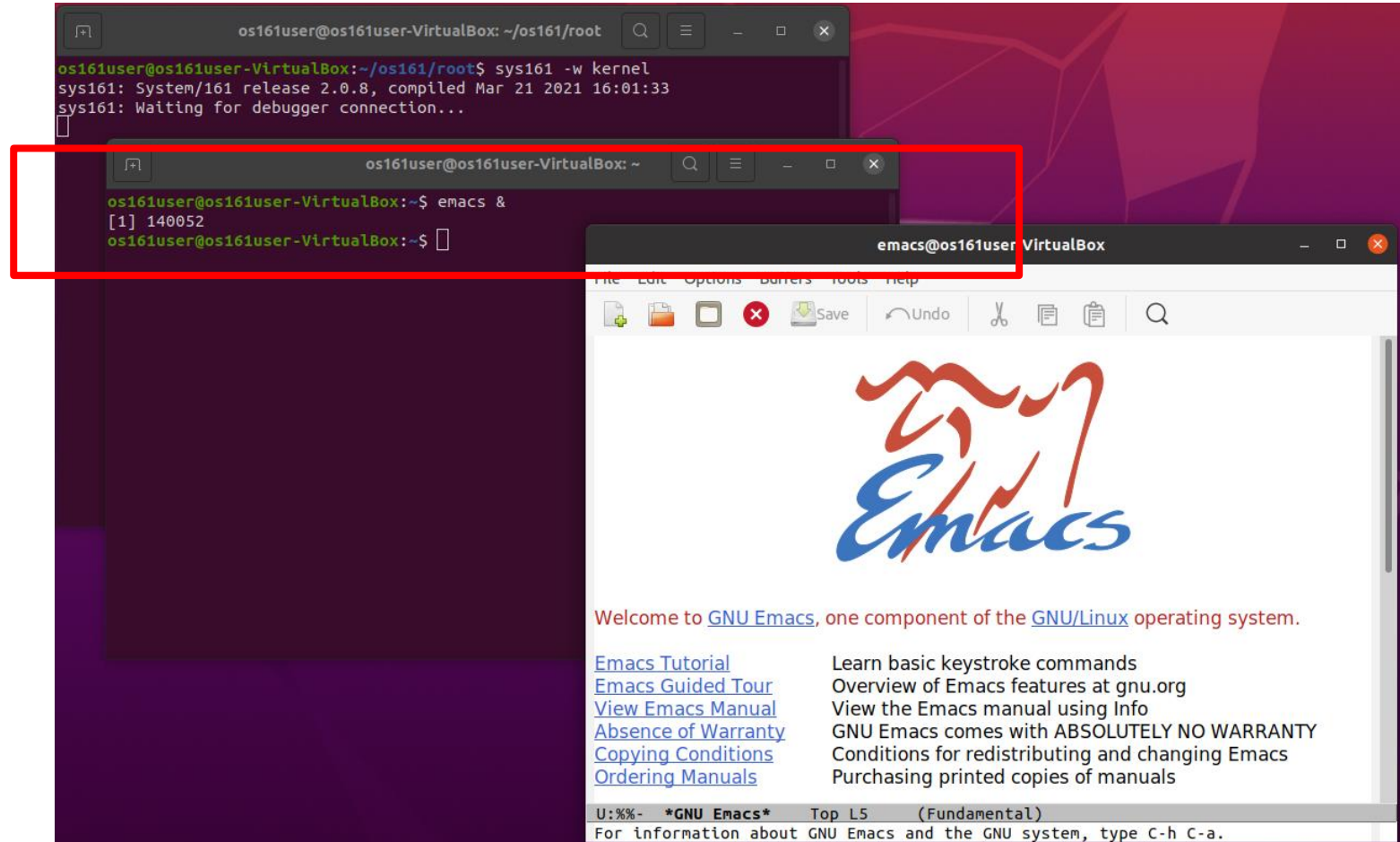
Making (building) OS161: Esequire Debugger

- Debug mode: Opening two terminals – Option 2
 - *sys11 -w kernel*
 - *ddd --debugger mips-Harvard-os161-gdb kernel*



Making (building) OS161: Esequire Debugger

- Debug mode: Opening two terminals – Option 3
 - `sys11 -w kernel`
 - `Emacs &`

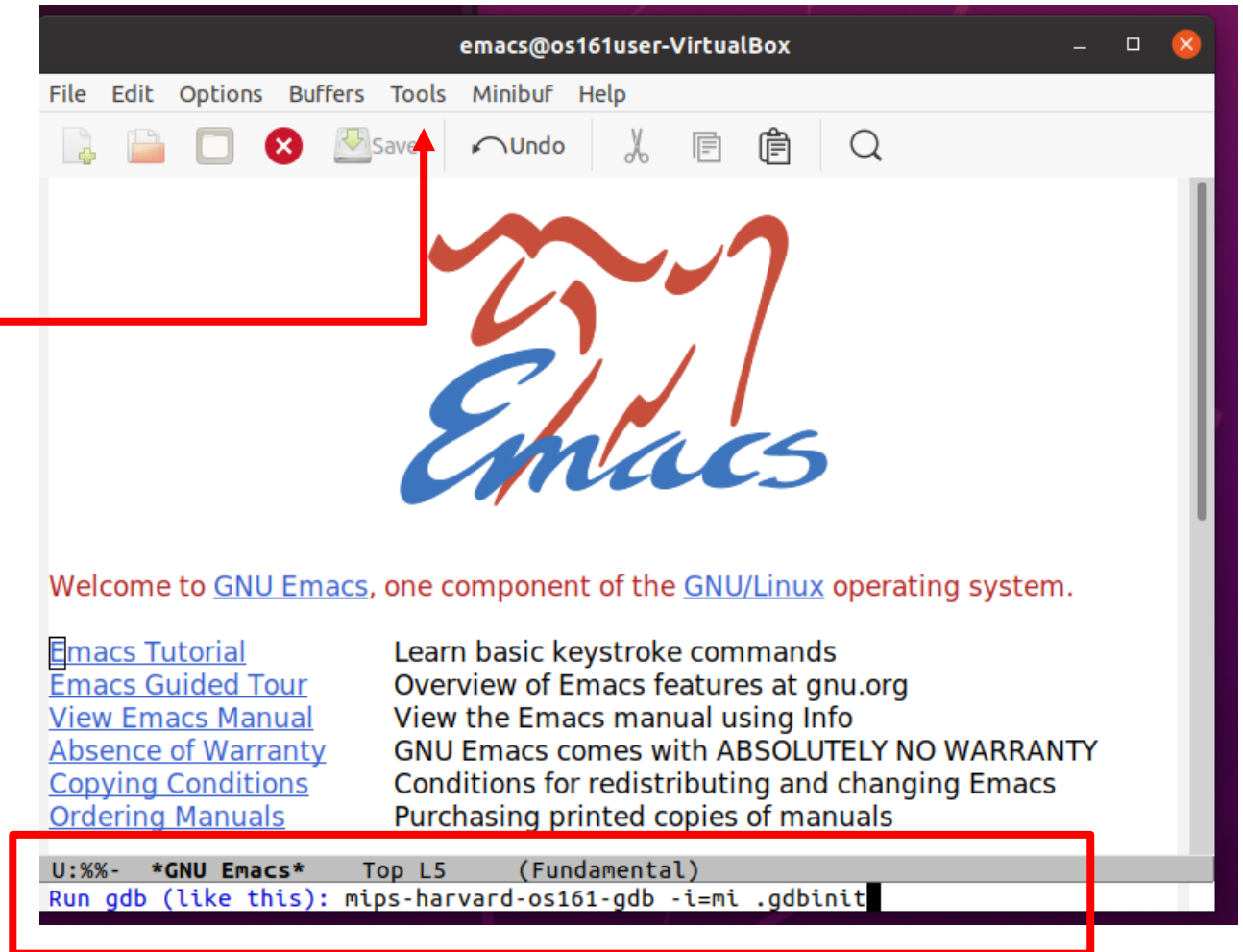


The screenshot shows a VirtualBox environment with three windows. The top window is a terminal titled 'os161user@os161user-VirtualBox: ~/os161/root' showing the command 'sys161 -w kernel' and its output: 'sys161: System/161 release 2.0.8, compiled Mar 21 2021 16:01:33' and 'sys161: Waiting for debugger connection...'. The middle window is a terminal titled 'os161user@os161user-VirtualBox: ~' with a red border, showing the command 'emacs &' and its output: '[1] 140052' and 'os161user@os161user-VirtualBox: ~'. The bottom window is the Emacs editor titled 'emacs@os161user-VirtualBox', displaying the Emacs logo and a welcome message: 'Welcome to GNU Emacs, one component of the GNU/Linux operating system.' It also includes links for 'Emacs Tutorial', 'Emacs Guided Tour', 'View Emacs Manual', 'Absence of Warranty', 'Copying Conditions', and 'Ordering Manuals', along with a list of resources: 'Learn basic keystroke commands', 'Overview of Emacs features at gnu.org', 'View the Emacs manual using Info', 'GNU Emacs comes with ABSOLUTELY NO WARRANTY', 'Conditions for redistributing and changing Emacs', and 'Purchasing printed copies of manuals'. The status bar at the bottom shows 'U:%%- *GNU Emacs* Top L5 (Fundamental)' and 'For information about GNU Emacs and the GNU system, type C-h C-a.'

Making (building) OS161: Esegueire Debugger

- Debug mode: Opening two terminals – Option 3
 - `sys11 -w kernel`
 - *Emacs &*

Tools >> Debugger



Modifying to: `mips-Harvard-os161 -i=mi .gdbinit`

Making (building) OS161: Eseguire Debugger

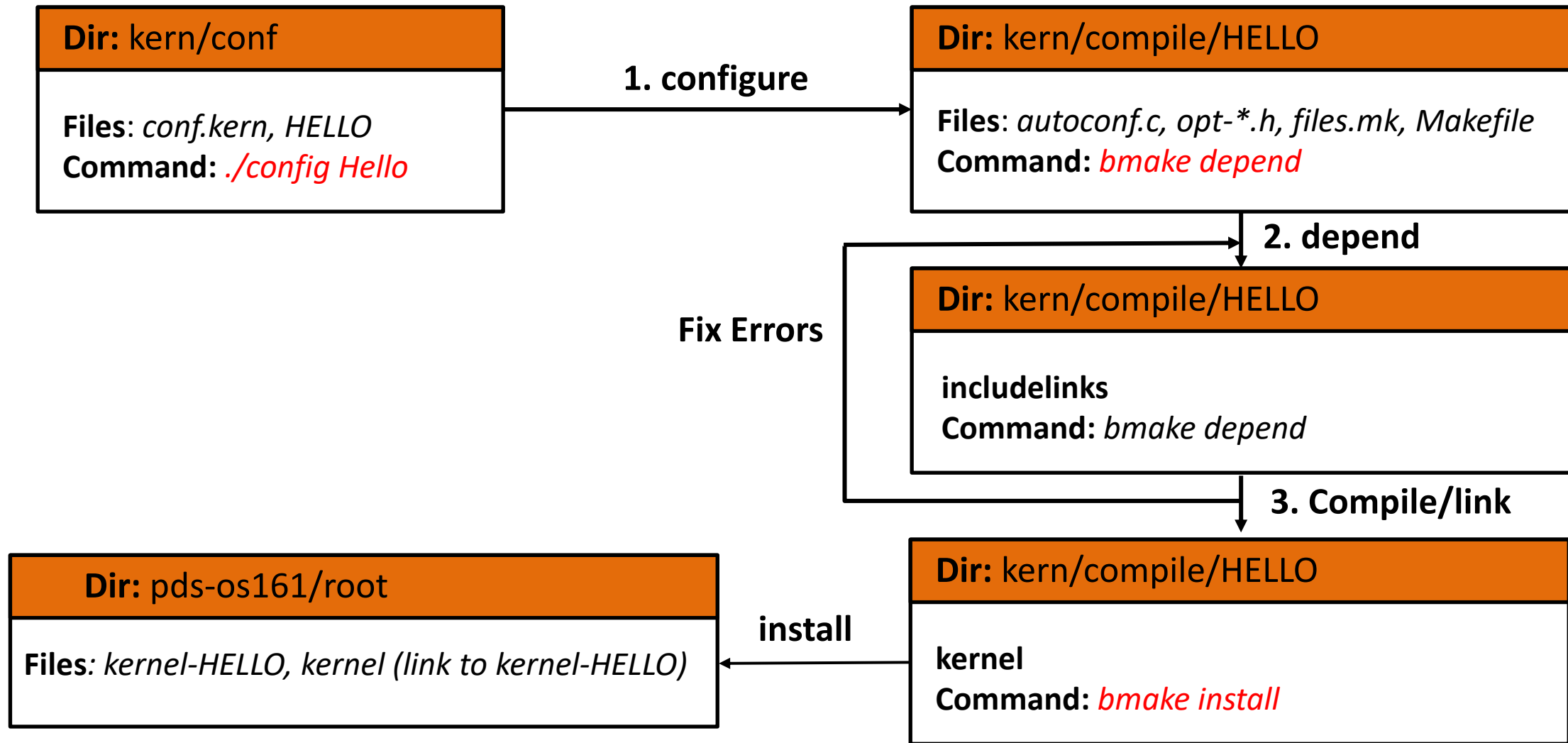
- Debug mode: Opening two terminals – Option 3
 - `sys11 -w kernel`
 - *Emacs* &



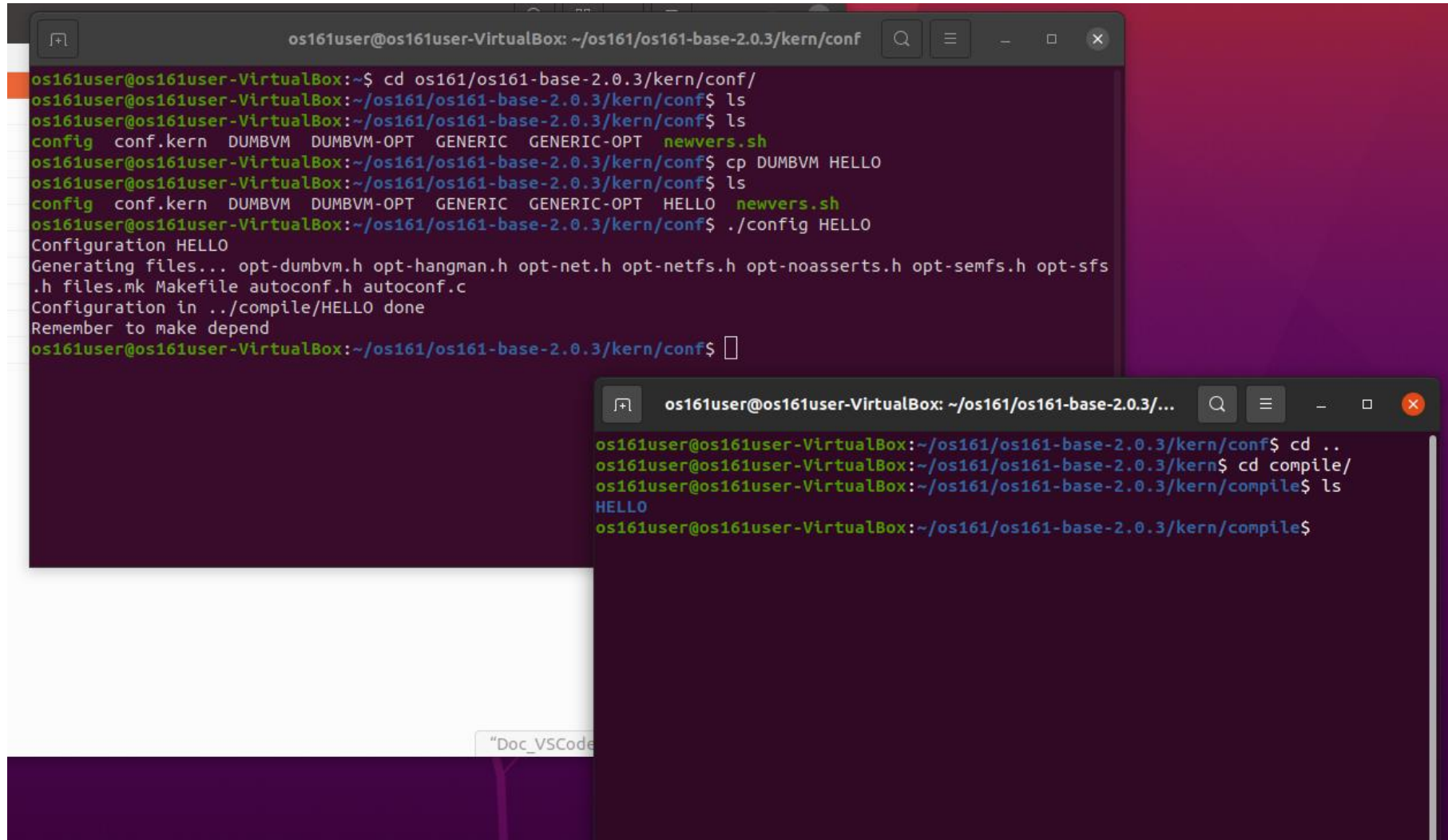
Making (building) OS161 new release: HELLO

- Three main directories:
 - Home/os161/os161-base-2.03/kern/conf
 - Home/os161/os161-base-2.03/kern/compile
 - Home/os161/root

Making (building) OS161 new release: HELLO



Making (building) OS161 new release: HELLO



The image shows two terminal windows from a VirtualBox environment, demonstrating the steps to build the OS161 HELLO kernel. The top window shows the configuration process, and the bottom window shows the compilation process.

```
os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/kern/conf
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ cd os161/os161-base-2.0.3/kern/conf/
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ ls
config  conf.kern  DUMBVM  DUMBVM-OPT  GENERIC  GENERIC-OPT  newvers.sh
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ cp DUMBVM HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ ls
config  conf.kern  DUMBVM  DUMBVM-OPT  GENERIC  GENERIC-OPT  HELLO  newvers.sh
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ ./config HELLO
Configuration HELLO
Generating files... opt-dumbvm.h opt-hangman.h opt-net.h opt-netfs.h opt-noasserts.h opt-semfs.h opt-sfs
.h files.mk Makefile autoconf.h autoconf.c
Configuration in ../compile/HELLO done
Remember to make depend
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$

os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/...
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ cd ..
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern$ cd compile/
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile$ ls
HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile$
```

Making (building) OS161 new release: HELLO

The image shows the process of building the OS161 'HELLO' release. It consists of two terminal windows and a file browser view.

Terminal 1 (Top): Shows the configuration process in the `~/os161/os161-base-2.0.3/kern/conf` directory.

```
os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/kern/conf
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ cd os161/os161-base-2.0.3/kern/conf/
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ ls
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ ls
config conf.kern DUMBVM DUMBVM-OPT GENERIC GENERIC-OPT newvers.sh
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ cp DUMBVM HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ ls
config conf.kern DUMBVM DUMBVM-OPT GENERIC GENERIC-OPT HELLO newvers.sh
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ ./config HELLO
Configuration HELLO
Generating files... opt-dumbvm.h opt-hangman.h opt-net.h opt-netfs.h opt-noasserts.h opt-semfs.h opt-sfs
.h files.mk Makefile autoconf.h autoconf.c
Configuration in ../compile/HELLO done
Remember to make depend
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$
```

Terminal 2 (Bottom): Shows the compilation process in the `~/os161/os161-base-2.0.3/kern/compile/HELLO` directory.

```
os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/kern/compile/HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ cd ..
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern$ cd compile/
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile$ ls
HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile$ ls HELLO/
autoconf.c includelinks opt-hangman.h opt-noasserts.h
autoconf.h Makefile opt-netfs.h opt-semfs.h
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$ bmake depend
```

File Browser (Bottom): Shows the directory structure of the build. The breadcrumb path is `Home > os161 > os161-base-2.0.3 > kern > compile`. The file list shows:

Name	Size	Modified
HELLO	132 items	16:54
.keep_me	0 bytes	24 gen 2017

A red arrow points from the `bmake depend` command in the terminal to the `HELLO` directory entry in the file browser.

Making (building) OS161 new release: HELLO

The image shows two overlapping file manager windows. The top window displays the contents of the 'os161' directory, specifically the 'os161-base-2.0.3' subdirectory. The bottom window shows the 'os161-base-2.0.3' directory, highlighting the 'HELLO' folder. A red arrow indicates the relationship between the 'includelinks' folder in the top window and the 'HELLO' folder in the bottom window.

Top Window: os161 / os161-base-2.0.3

Name	Size	Modified
includelinks	5 items	16:51
autoconf.c	8,3 kB	16:47
autoconf.h	1,8 kB	16:47
files.mk	4,8 kB	16:47
Makefile	306 bytes	16:47
opt-dumbvm.h	138 bytes	16:47
opt-hangman.h	142 bytes	16:47
opt-net.h	126 bytes	16:47
opt-netfs.h	134 bytes	16:47
opt-noasserts.h	150 bytes	16:47
opt-semfs.h	134 bytes	16:47
opt-sfs.h	126 bytes	16:47
.depend	52,8 kB	16:54
.depend.adddi3.c	354 bytes	16:54
.depend.anddi3.c	354 bytes	16:54
.depend.array.c	258 bytes	16:54
.depend.arraytest.c	258 bytes	16:54
.depend.ashldi3.c	356 bytes	16:54
.depend.ashrdi3.c	356 bytes	16:54
.depend.atoi.c	222 bytes	16:54
.depend.autoconf.c	253 bytes	16:54
.depend.beep.c	278 bytes	16:54
.depend.beep_ltimer.c	292 bytes	16:54
.depend.bitmap.c	261 bytes	16:54
.depend.bitmaptest.c		

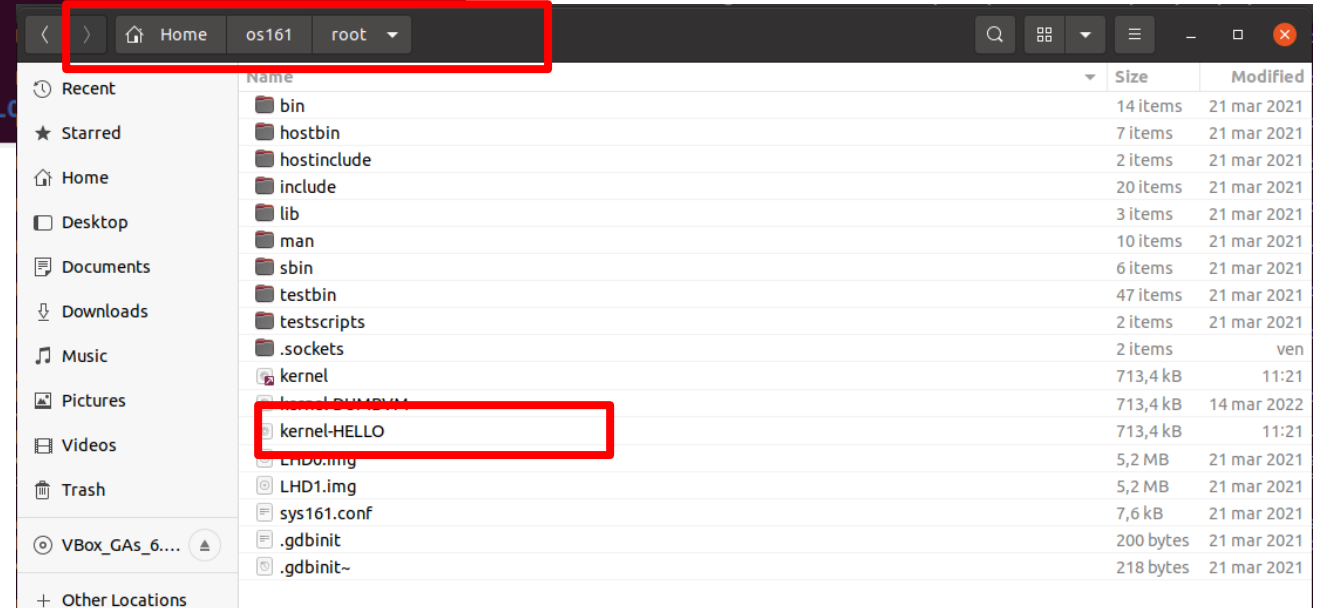
Bottom Window: os161-base-2.0.3

Name	Size	Modified
HELLO	132 items	16:54
.keep_me	0 bytes	24 gen 2017

[illegible]

Making (building) OS161 new release: HELLO

```
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$  
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$  
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$  
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$  
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$  
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$ ls kernel  
kernel  
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$ ls -la kernel  
ls: cannot access '-': No such file or directory  
ls: cannot access 'la': No such file or directory  
kernel  
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$ ls -la kernel  
-rwxrwxr-x 1 os161user os161user 713426 dic 18 11:19 kernel  
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$ bmake install  
[ -d /home/os161user/os161/root ] || mkdir /home/os161user/os161/root  
cp kernel /home/os161user/os161/root/kernel-HELLO  
rm -f /home/os161user/os161/root/kernel  
ln -s kernel-HELLO /home/os161user/os161/root/kernel  
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
```



Making (building)

Code browsing/understanding

- Edit .c/.h files in `os161/os161-base-2.0.2/kern`
- Use browsable code from `os161_doc/os161/html/index.html` Kernel configuration/options
- `os161/os161-base-2.0.2/kern/conf/conf.kern`: definition of options and list of files
- 4 kernel configurations already available: DUMBVM, DUMBVM-OPT, GENERIC, GENERIC-OPT (they include `conf.kern`).
- To generate a new configuration, copy and modify: e.g. HELLO (new configuration) copied from GENERIC and modified
- `COMMAND` (in `os161/os161-base-2.0.2/kern/conf`)
 - `./config HELLO`
 - Generates `os161/os161-base-2.0.2/kern/compile/HELLO`

Making (building)

Compilation/make

- In os161/os161-base-2.0.2/kern/compile/HELLO (or equivalent directory)
- Make dependencies: scan C files and generate rules to (automatically) recompile a given source C file (generate the object file) if a .h is modified
- bmake depend
- Compile (build executable: e.g. kernel-HELLO)
 - bmake
 - if compilation errors, correct code and rerun
- Install (copy) executable in pds-os161/root
 - bmake install
 - Copies kernel-HELLO (or other) and generates symbolic link “kernel”

Running/Debugging

Work in pds-os161/root

MIPS virtual machine (sys161) configured in sys161.conf

- One important line to be properly edited
 - mainboard ramsize=1024K cpus=1
- Running (bootstrap) kernel on mips machine
 - sys161 kernel (without debugger support)
 - sys161 -w kernel (with debugger support: waiting for debugger connection on socket)
 - ... or other

OS161 kernel

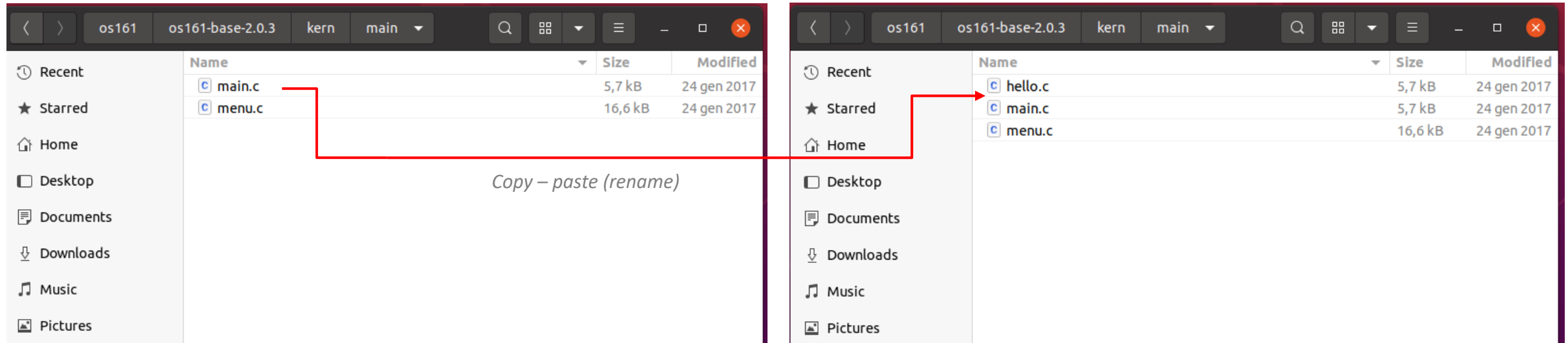
Kernel main (kmain)

- os161/os161-base-2.0.2/kern/main
- main.c, menu.c
- Basic support (partial): threads, memory management, system calls, semaphores, running user executable (ELF format)

```
void kmain(char *arguments) {  
    boot();  
    menu(arguments);  
    /* Should not get here */  
}
```

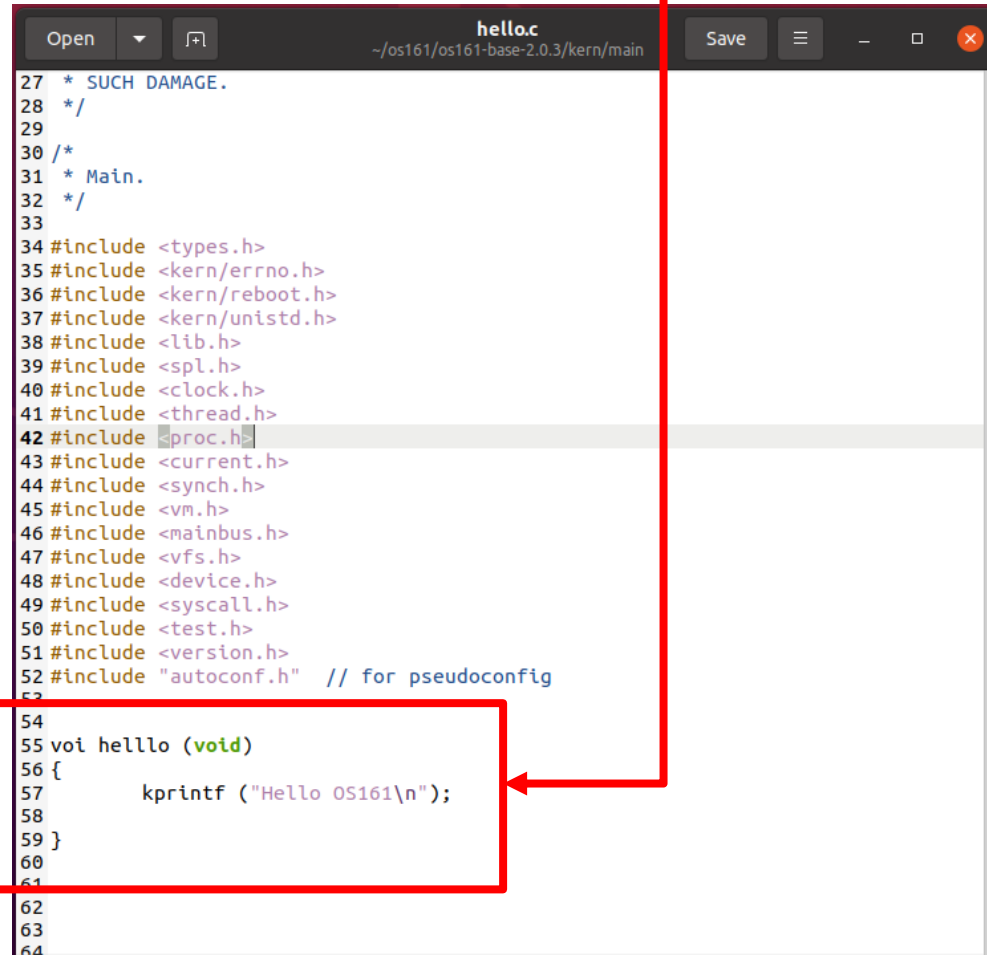
OS161 kernel – Lab01

- Adding `kprintf()` to hello

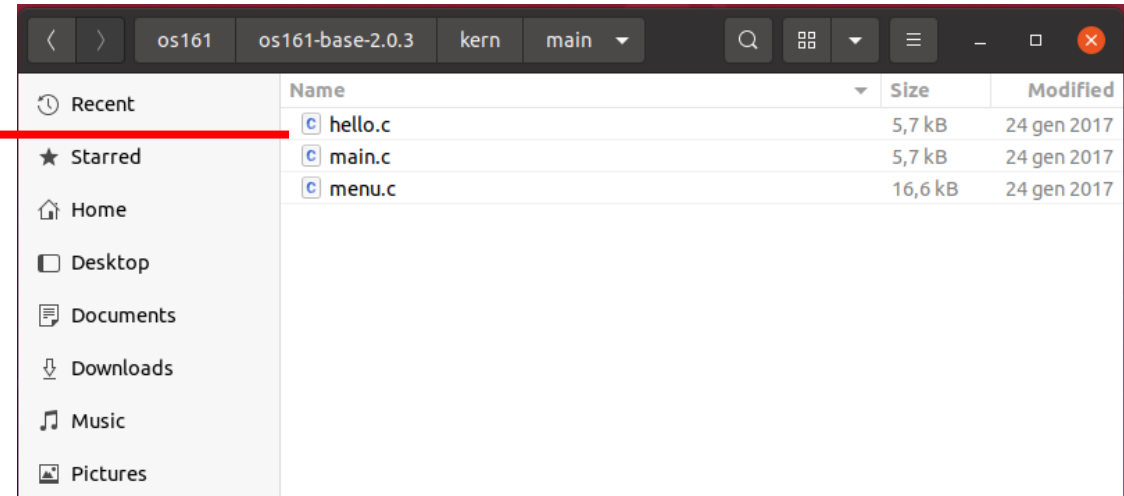


OS161 kernel – Lab01

- Adding *kprintf()* to hello

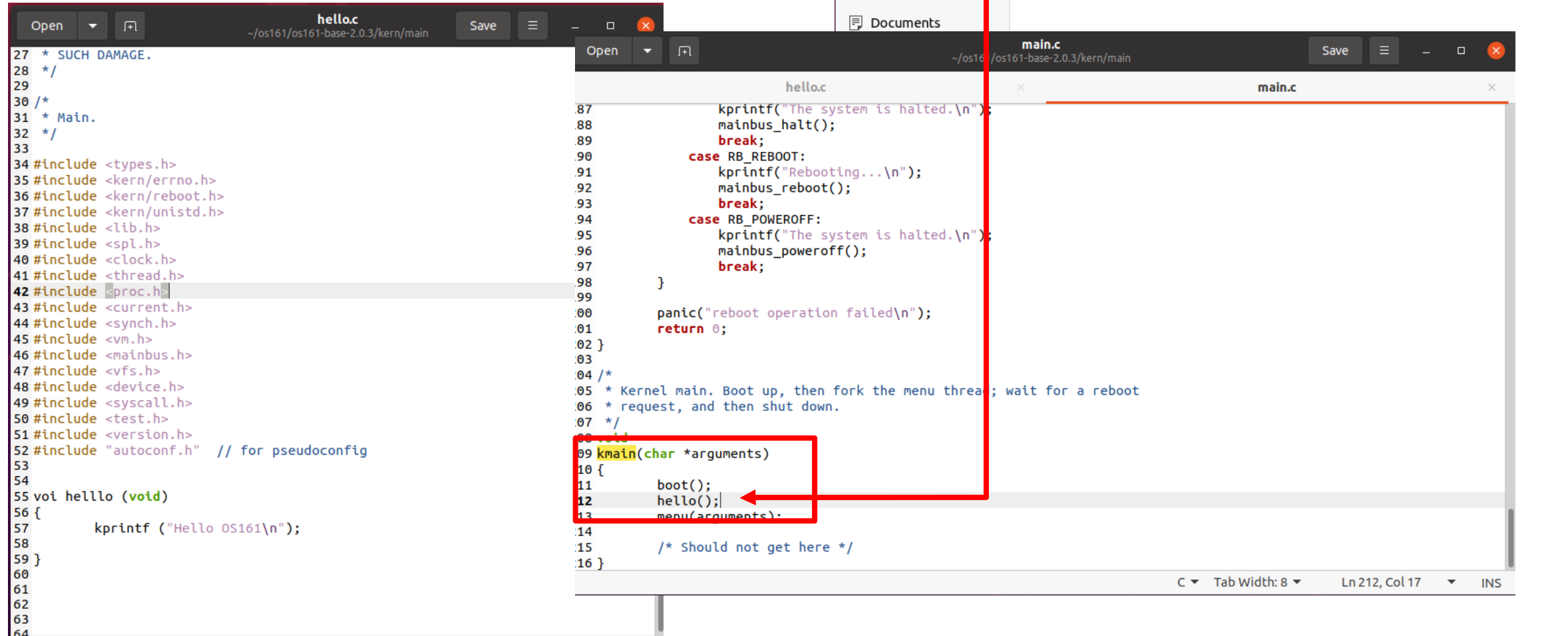


```
27 * SUCH DAMAGE.
28 */
29
30 /*
31 * Main.
32 */
33
34 #include <types.h>
35 #include <kern/errno.h>
36 #include <kern/reboot.h>
37 #include <kern/unistd.h>
38 #include <lib.h>
39 #include <spl.h>
40 #include <clock.h>
41 #include <thread.h>
42 #include <proc.h>
43 #include <current.h>
44 #include <synch.h>
45 #include <vm.h>
46 #include <mainbus.h>
47 #include <vfs.h>
48 #include <device.h>
49 #include <syscall.h>
50 #include <test.h>
51 #include <version.h>
52 #include "autoconf.h" // for pseudoconfig
53
54
55 void hello (void)
56 {
57     kprintf ("Hello OS161\n");
58 }
59
60
61
62
63
64
```



OS161 kernel – Lab01

- Adding `kprintf()` to hello



The screenshot displays the OS161 kernel development environment. At the top, a file browser window shows the directory structure: `os161` / `os161-base-2.0.3` / `kern` / `main`. It lists files `hello.c` (5,7 kB), `main.c` (5,7 kB), and `menu.c` (16,6 kB), all modified on 24 gen 2017.

Below the file browser, two code editors are open. The left editor, titled `hello.c`, shows the following code:

```
27 * SUCH DAMAGE.
28 */
29
30 /*
31 * Main.
32 */
33
34 #include <types.h>
35 #include <kern/errno.h>
36 #include <kern/reboot.h>
37 #include <kern/unistd.h>
38 #include <lib.h>
39 #include <spl.h>
40 #include <clock.h>
41 #include <thread.h>
42 #include <proc.h>
43 #include <current.h>
44 #include <synch.h>
45 #include <vm.h>
46 #include <mainbus.h>
47 #include <vfs.h>
48 #include <device.h>
49 #include <syscall.h>
50 #include <test.h>
51 #include <version.h>
52 #include "autoconf.h" // for pseudoconfig
53
54
55 void hello(void)
56 {
57     kprintf("Hello OS161\n");
58 }
59
60
61
62
63
64
```

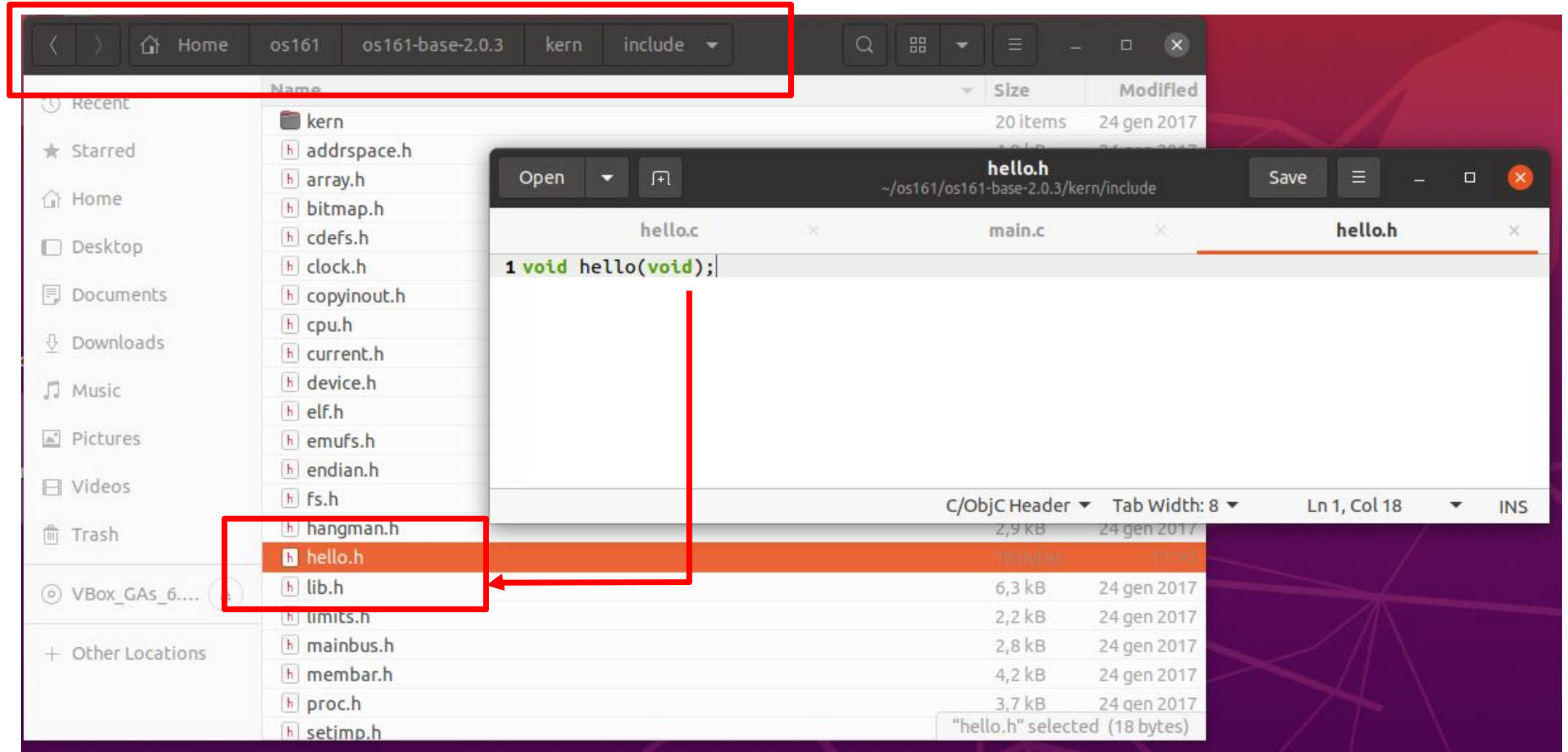
The right editor, titled `main.c`, shows the following code:

```
87 kprintf("The system is halted.\n");
88 mainbus_halt();
89 break;
90 case RB_REBOOT:
91     kprintf("Rebooting...\n");
92     mainbus_reboot();
93     break;
94 case RB_POWEROFF:
95     kprintf("The system is halted.\n");
96     mainbus_poweroff();
97     break;
98 }
99
100 panic("reboot operation failed\n");
101 return 0;
102 }
103
104 /*
105 * Kernel main. Boot up, then fork the menu thread; wait for a reboot
106 * request, and then shut down.
107 */
108 void
109 kmain(char *arguments)
110 {
111     boot();
112     hello();
113     menu(arguments);
114
115     /* Should not get here */
116 }
```

A red box highlights the `kmain` function signature and its body in `main.c`. A red arrow points from the `hello()` call on line 112 to the `hello.c` editor, indicating the source of the function.

OS161 kernel – Lab01

- Adding `kprintf()` to hello – Creating a file `hello.h`



OS161 kernel – Lab01

- Adding `kprintf()` to hello – Adding `hello.h` to `hello.c` and `main.c`

The image shows two side-by-side code editors. The left editor is titled 'hello.c' and the right is titled 'main.c'. Both are located at the path '~/os161/os161-base-2.0.3/kern/main'. In the 'hello.c' editor, lines 52 and 53 are highlighted with a red box: `#include "autoconf.h" // for pseudoconfig` and `#include "hello.h"`. A red arrow points from the 'hello.h' include line to the 'main.c' editor. In the 'main.c' editor, lines 52 and 53 are also highlighted with a red box: `#include "autoconf.h" // for pseudoconfig` and `#include "hello.h"`. The 'main.c' editor also shows a red box around line 37: `#include <kern/unistd.h>`. The status bar at the bottom of the editors shows 'C Tab Width: 8' and 'Ln 56, Col 3'.

```
hello.c
41 #include <thread.h>
42 #include <proc.h>
43 #include <current.h>
44 #include <synch.h>
45 #include <vm.h>
46 #include <mainbus.h>
47 #include <vfs.h>
48 #include <device.h>
49 #include <syscall.h>
50 #include <test.h>
51 #include <version.h>
52 #include "autoconf.h" // for pseudoconfig
53 #include "hello.h"
54
55 void hello(void)
56 {
57     kprintf("Hello OS161\n");
58 }
59
60
61
62
63
64
```

```
main.c
37 #include <kern/unistd.h>
38 #include <lib.h>
39 #include <spl.h>
40 #include <clock.h>
41 #include <thread.h>
42 #include <proc.h>
43 #include <current.h>
44 #include <synch.h>
45 #include <vm.h>
46 #include <mainbus.h>
47 #include <vfs.h>
48 #include <device.h>
49 #include <syscall.h>
50 #include <test.h>
51 #include <version.h>
52 #include "autoconf.h" // for pseudoconfig
53 #include "hello.h"
54
55
56 /*
57 * These two pieces of data are maintained by the makefiles and build system.
58 * buildconfig is the name of the config file the kernel was configured with.
59 * buildversion starts at 1 and is incremented every time you link a kernel.
60 *
61 * The purpose is not to show off how many kernels you've linked, but
62 * to make it easy to make sure that the kernel you just tested is the
```

OS161 kernel – Lab01

- Adding `kprintf()` to hello – Adding hello as an option

```
HELLO
~/os161/os161-base-2.0.3/kern/conf

12 #
13 device lamebus0          # System/161 main bus
14 device emu* at lamebus*  # Emulator passthrough
15 device ltrace* at lamebus* # trace161 trace control device
16 device ltimer* at lamebus* # Timer device
17 device lrandom* at lamebus* # Random device
18 device lhd* at lamebus*   # Disk device
19 device lser* at lamebus*  # Serial port
20 #device lscreen* at lamebus* # Text screen (not supported yet)
21 #device lnet* at lamebus*  # Network interface (not supported yet)
22 device beep0 at ltimer*   # Abstract beep handler
23 device con0 at lser*      # Abstract console on serial port
24 #device con0 at lscreen*  # Abstract console on screen (not supported yet)
25 device rtclock0 at ltimer* # Abstract realtime clock
26 device random0 at lrandom* # Abstract randomness device
27
28 #options net              # Network stack (not supported yet)
29 options semfs             # Semaphores for userland
30
31 options sfs               # Always use the file system
32 #options netfs            # You might write this as a probe
33
34 options dumbvm            # Chewing gum and baling wire.
35 options hello             #hello
36
37
38
```

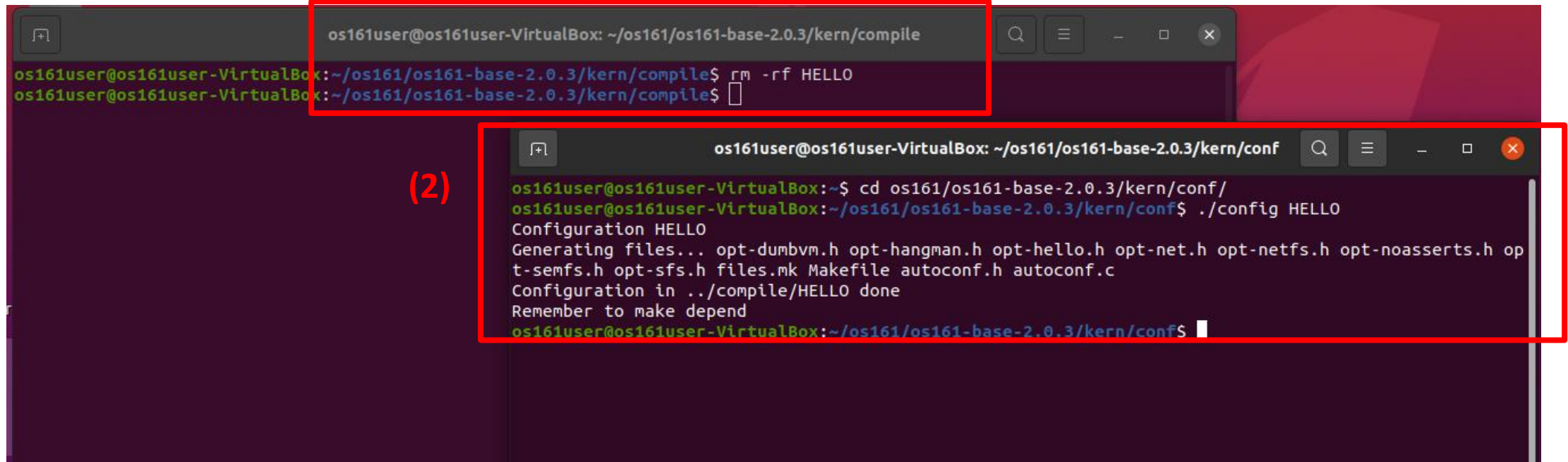
```
conf.kern
~/os161/os161-base-2.0.3/kern/conf

424 # Note that errors are completely contained in the emu device.
425 #
426
427
428 #####
429 #
430 #           Test code
431 #
432 #####
433
434 file          test/arraytest.c
435 file          test/btmaptest.c
436 file          test/threadlisttest.c
437 file          test/threadtest.c
438 file          test/tt3.c
439 file          test/synchtest.c
440 file          test/semunit.c
441 file          test/kmalloc_test.c
442 file          test/fstest.c
443 optfile net   test/nettest.c
444
445 # NEW PDS WORK
446 defoption hello
447 optfile hello main/hello.c
448
449
450
```

OS161 kernel – Lab01

- Adding `kprintf()` to hello – Build again

(1)

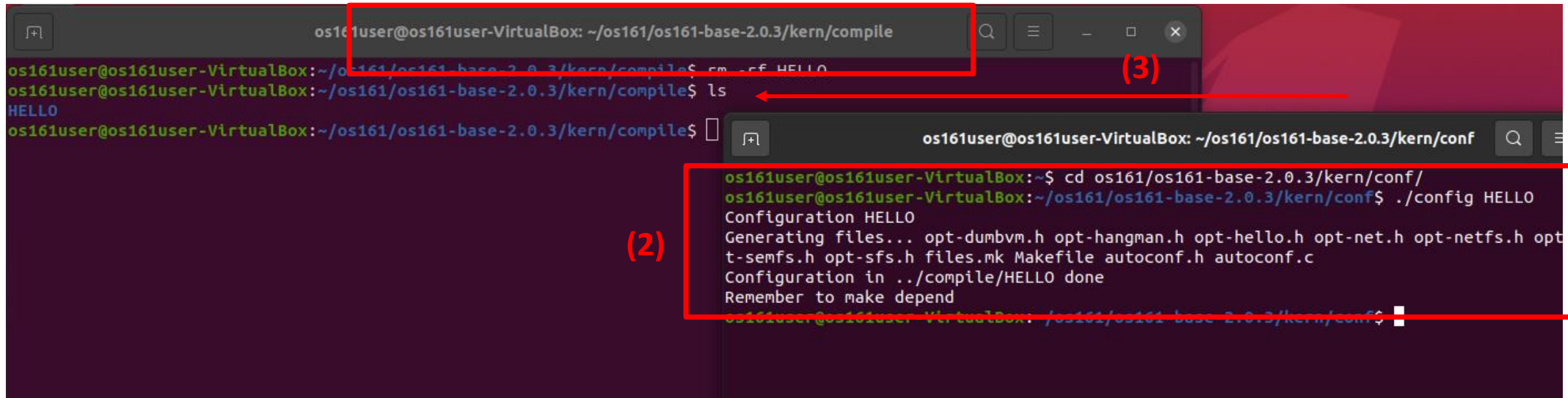


```
os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/kern/compile
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile$ rm -rf HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile$

(2)
os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/kern/conf
os161user@os161user-VirtualBox:~$ cd os161/os161-base-2.0.3/kern/conf/
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ ./config HELLO
Configuration HELLO
Generating files... opt-dumbvm.h opt-hangman.h opt-hello.h opt-net.h opt-netfs.h opt-noasserts.h opt-semfs.h opt-sfs.h files.mk Makefile autoconf.h autoconf.c
Configuration in ../compile/HELLO done
Remember to make depend
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$
```


OS161 kernel – Lab01

- Adding `kprintf()` to hello – Build again



The image shows two terminal windows from a VirtualBox environment. The left window is titled `os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/kern/compile` and shows the following commands and output:

```
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile$ rm -rf HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile$ ls
HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile$
```

The right window is titled `os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/kern/conf` and shows the following commands and output:

```
os161user@os161user-VirtualBox:~$ cd os161/os161-base-2.0.3/kern/conf/
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$ ./config HELLO
Configuration HELLO
Generating files... opt-dumbvm.h opt-hangman.h opt-hello.h opt-net.h opt-netfs.h opt-
t-semfs.h opt-sfs.h files.mk Makefile autoconf.h autoconf.c
Configuration in ../compile/HELLO done
Remember to make depend
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/conf$
```

Red annotations are present: (1) points to the terminal title bar of the left window; (2) points to the `./config HELLO` command in the right window; (3) points to the `ls` command in the left window.

OS161 kernel – Lab01

- Adding `kprintf()` to hello – Build again

```
os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/kern/compile/HELLO
os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/kern/compile$ rm -rf HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile$ ls
HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile$ cd HELLO/
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$ bmake depend
bmake includelinks
mkdir includelinks
mkdir includelinks/kern
ln -s ../../../../arch/mips/include includelinks/mips
ln -s ../../../../arch/mips/include/kern includelinks/kern/mips
ln -s ../../../../arch/sys161/include includelinks/sys161
ln -s mips includelinks/machine
ln -s mips includelinks/kern/machine
ln -s sys161 includelinks/platform
rm -f .depend.* || true
bmake realdepend
mips-harvard-os161-gcc -g -Og -Wall -W -Wwrite-strings -Wmissing-prototypes -Werror -std=gnu99 -mno-abicalls -fno-pic -ffixed-23 -nostdinc -I../../include -I../../dev -I. -Iincludelinks -ffreestanding -D_KERNEL -M ../../../../common/libc/p
rintf/__printf.c > .depend.__printf.c
mips-harvard-os161-gcc -g -Og -Wall -W -Wwrite-strings -Wmissing-prototypes -Werror -std=gnu99 -mno-abicalls -fno-pic -ffixed-23 -nostdinc -I../../include -I../../dev -I. -Iincludelinks -ffreestanding -D_KERNEL -M ../../../../common/libc/p
rintf/snprintf.c > .depend.snprintf.c
mips-harvard-os161-gcc -g -Og -Wall -W -Wwrite-strings -Wmissing-prototypes -Werror -std=gnu99 -mno-abicalls -fno-pic -ffixed-23 -nostdinc -I../../include -I../../dev -I. -Iincludelinks -ffreestanding -D_KERNEL -M ../../../../common/libc/s
tdlib/atoi.c > .depend.atoi.c
```

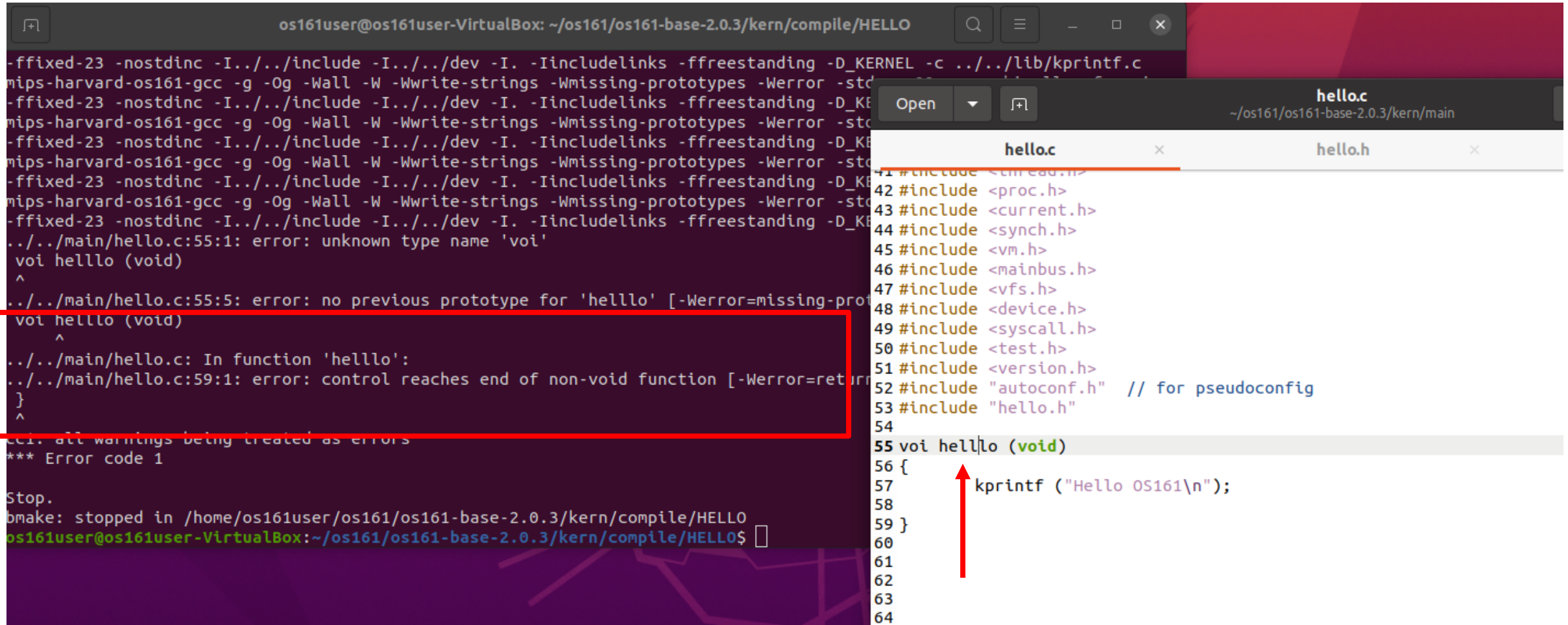
OS161 kernel – Lab01

- Adding `kprintf()` to hello – Build again

```
os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/kern/compile/HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$ bmake
mips-harvard-os161-gcc -g -Og -Wall -W -Wwrite-strings -Wmissing-prototypes -Werror -std=gnu99 -mno-abicalls -fno-pic
-ffixed-23 -nostdinc -I../include -I../dev -I. -Iincludelinks -ffreestanding -D_KERNEL -c ../../common/libc/p
rintf/__printf.c
mips-harvard-os161-gcc -g -Og -Wall -W -Wwrite-strings -Wmissing-prototypes -Werror -std=gnu99 -mno-abicalls -fno-pic
-ffixed-23 -nostdinc -I../include -I../dev -I. -Iincludelinks -ffreestanding -D_KERNEL -c ../../common/libc/p
rintf/snprintf.c
mips-harvard-os161-gcc -g -Og -Wall -W -Wwrite-strings -Wmissing-prototypes -Werror -std=gnu99 -mno-abicalls -fno-pic
-ffixed-23 -nostdinc -I../include -I../dev -I. -Iincludelinks -ffreestanding -D_KERNEL -c ../../common/libc/s
tdlib/atoi.c
mips-harvard-os161-gcc -g -Og -Wall -W -Wwrite-strings -Wmissing-prototypes -Werror -std=gnu99 -mno-abicalls -fno-pic
-ffixed-23 -nostdinc -I../include -I../dev -I. -Iincludelinks -ffreestanding -D_KERNEL -c ../../common/libc/s
tring/bzero.c
mips-harvard-os161-gcc -g -Og -Wall -W -Wwrite-strings -Wmissing-prototypes -Werror -std=gnu99 -mno-abicalls -fno-pic
-ffixed-23 -nostdinc -I../include -I../dev -I. -Iincludelinks -ffreestanding -D_KERNEL -c ../../common/libc/s
tring/memcpy.c
mips-harvard-os161-gcc -g -Og -Wall -W -Wwrite-strings -Wmissing-prototypes -Werror -std=gnu99 -mno-abicalls -fno-pic
-ffixed-23 -nostdinc -I../include -I../dev -I. -Iincludelinks -ffreestanding -D_KERNEL -c ../../common/libc/s
tring/memmove.c
```

OS161 kernel – Lab01

- Adding `kprintf()` to hello – Error Found



The screenshot shows a terminal window on the left and a code editor on the right. The terminal window displays the output of a compilation command, which includes several error messages. A red box highlights the following errors:

```
../main/hello.c:55:1: error: unknown type name 'voi'
voi hello (void)
^
../main/hello.c:55:5: error: no previous prototype for 'hello' [-Werror=missing-proto
voi hello (void)
^
../main/hello.c: In function 'hello':
../main/hello.c:59:1: error: control reaches end of non-void function [-Werror=return
}
^
cc1: all warnings being treated as errors
*** Error code 1

Stop.
bmake: stopped in /home/os161user/os161/os161-base-2.0.3/kern/compile/HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
```

The code editor on the right shows the `hello.c` file. A red arrow points to the `kprintf` function call on line 57:

```
55 voi hello (void)
56 {
57     kprintf ("Hello OS161\n");
58 }
59
60
61
62
63
64
```


OS161 kernel – Lab01

- Adding `kprintf()` to hello – Error Found >> Fixing and bmake again

```
os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/kern/compile/HELLO
-ffixed-23 -nostdinc -I../include -I../dev -I. -Iincludelinks -ffreestanding -D_KERNEL -c ../main/hello.c
../main/hello.c:55:1: error: unknown type name 'voi'
voi hello (void)
^
../main/hello.c:55:5: error: no previous prototype for 'hello' [-Werror=missing-prototypes]
voi hello (void)
^
../main/hello.c: In function 'helllo':
../main/hello.c:59:1: error: control reaches end of non-void function [-Werror=return-type]
}
^
cc1: all warnings being treated as errors
*** Error code 1

Stop.
bmake: stopped in /home/os161user/os161/os161-base-2.0.3/kern/compile/HELLO
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$ bmake
mips-harvard-os161-gcc -g -Og -Wall -W -Wwrite-strings -Wmissing-prototypes -Werror -std=gnu99 -mno-abicalls -fno-pic
-ffixed-23 -nostdinc -I../include -I../dev -I. -Iincludelinks -ffreestanding -D_KERNEL -c ../main/hello.c
../main/hello.c:55:1: error: unknown type name 'voi'
```

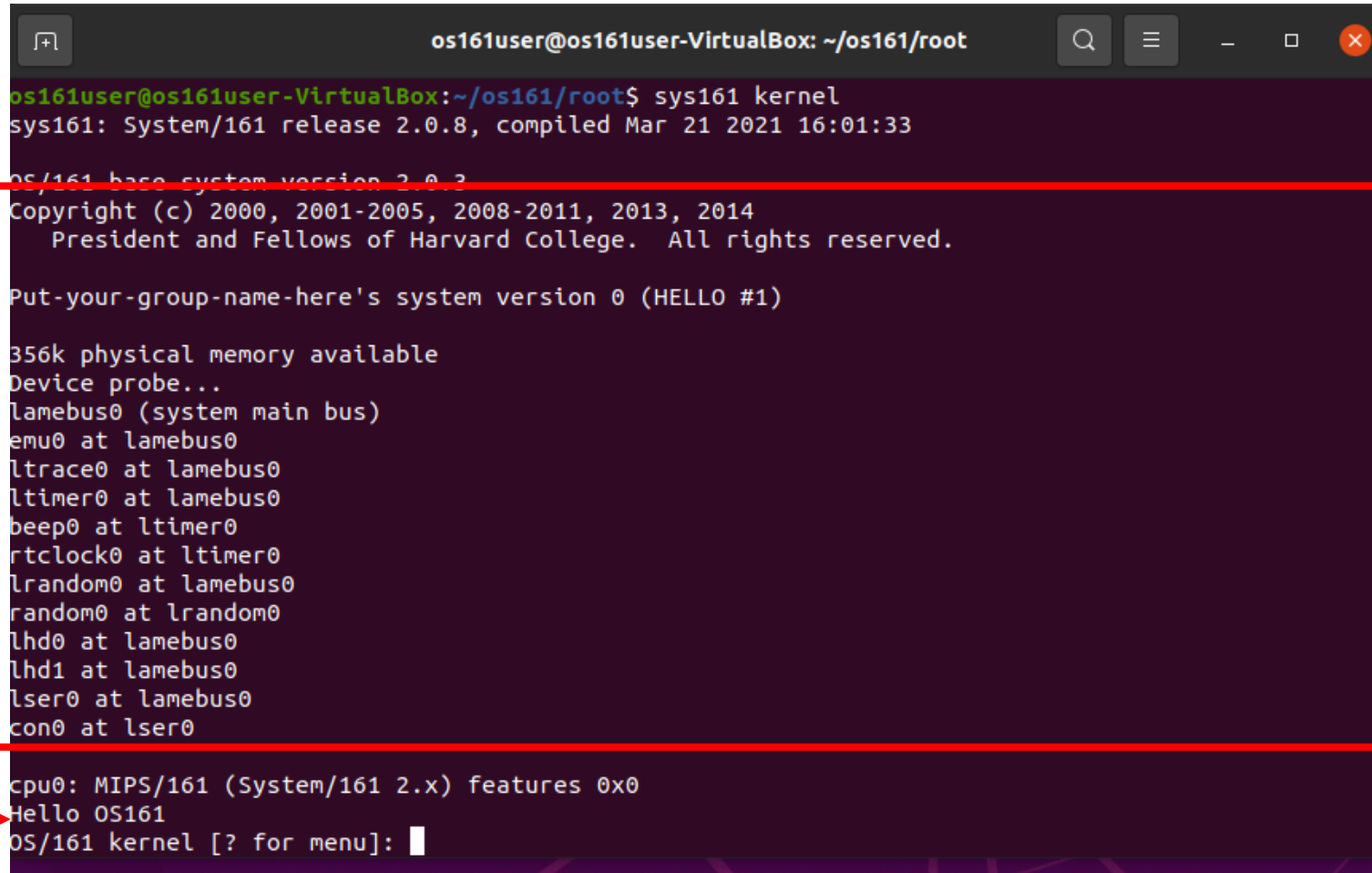
OS161 kernel – Lab01

- Adding `kprintf()` to hello – `bmake install`

```
os161user@os161user-VirtualBox: ~/os161/os161-base-2.0.3/kern/compile/HELLO
ve.o memset.o strcat.o strchr.o strcmp.o strcpy.o strlen.o strrchr.o strtok_r.o autoconf.o beep.o console.o random.o r
tclock.o beep_ltimer.o con_lser.o emu_att.o emu.o lamebus.o lhd_att.o lhd.o lrandom_att.o lrandom.o lser_att.o lser.o
ltimer_att.o ltimer.o ltrace_att.o ltrace.o random_lrandom.o rtclock_ltimer.o semfs_fsops.o semfs_obj.o semfs_vnops.o
sfs_balloc.o sfs_bmap.o sfs_dir.o sfs_fsops.o sfs_inode.o sfs_io.o sfs_vnops.o array.o bitmap.o bswap.o kgets.o kprint
f.o misc.o time.o uio.o hello.o main.o menu.o proc.o loadelf.o runprogram.o time_syscalls.o arraytest.o bitmaptest.o f
stest.o kmalloc.o kmalloc.o semunit.o synchtest.o threadlisttest.o threadtest.o tt3.o clock.o spinlock.o spl.o synch.o threa
d.o threadlist.o device.o devnull.o vfscwd.o vfstail.o vfstest.o vfstest.o vnode.o kmalloc.o trap.o syscal
l.o cpu.o switchframe.o switch.o thread_machdep.o threadstart.o dumbvm.o ram.o adddi3.o anddi3.o ashldi3.o ashrdi3.o c
mpdi2.o divdi3.o iordi3.o lshldi3.o lshrdi3.o moddi3.o muldi3.o negdi2.o notdi2.o qdivrem.o subdi3.o ucmpdi2.o udivdi3
.o umoddi3.o xordi3.o setjmp.o copyinout.o cache-mips161.o exception-mips1.o tlb-mips161.o lamebus_machdep.o start.o v
ers.o -o kernel
*** This is HELLO build #1 ***
mips-harvard-os161-size kernel
  text    data    bss     dec      hex filename
 146120    192    6976  153288  256c8 kernel
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$ bmake install
[ -d /home/os161user/os161/root ] || mkdir /home/os161user/os161/root
cp kernel /home/os161user/os161/root/kernel-HELLO
rm -f /home/os161user/os161/root/kernel
ln -s kernel-HELLO /home/os161user/os161/root/kernel
os161user@os161user-VirtualBox:~/os161/os161-base-2.0.3/kern/compile/HELLO$
```

OS161 kernel – Lab01

- Adding `kprintf()` to hello – execute new kernel



```
os161user@os161user-VirtualBox: ~/os161/root
os161user@os161user-VirtualBox:~/os161/root$ sys161 kernel
sys161: System/161 release 2.0.8, compiled Mar 21 2021 16:01:33
OS/161 base system version 2.0.8
Copyright (c) 2000, 2001-2005, 2008-2011, 2013, 2014
  President and Fellows of Harvard College.  All rights reserved.

Put-your-group-name-here's system version 0 (HELLO #1)

356k physical memory available
Device probe...
lamebus0 (system main bus)
  emu0 at lamebus0
  ltrace0 at lamebus0
  ltimer0 at lamebus0
    beep0 at ltimer0
    rtclock0 at ltimer0
  lrandom0 at lamebus0
    random0 at lrandom0
  lhd0 at lamebus0
  lhd1 at lamebus0
  lser0 at lamebus0
  lcon0 at lser0

cpu0: MIPS/161 (System/161 2.x) features 0x0
Hello OS161
OS/161 kernel [? for menu]:
```