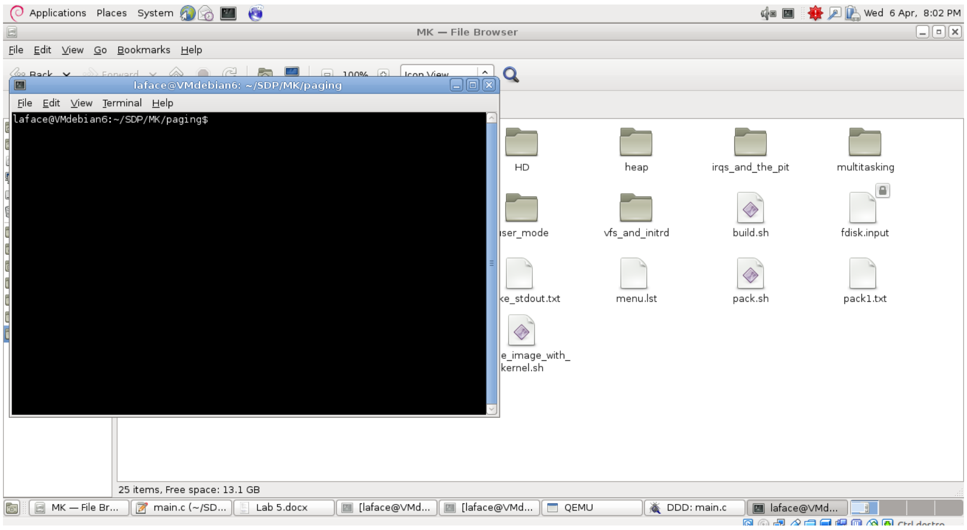
Guide to the lab5 (unofficial)

1. Finding the files to use

Once the virtual machine is installed, go to **/home/laface/SDP/MK**. All the files needed are there. For lab 5 the subfolder **paging** is used.

2. Build

Open a terminal and go to **/home/laface/SDP/MK/paging**



```
laface@VMdebian6:~/SDP/MK/paging$
```

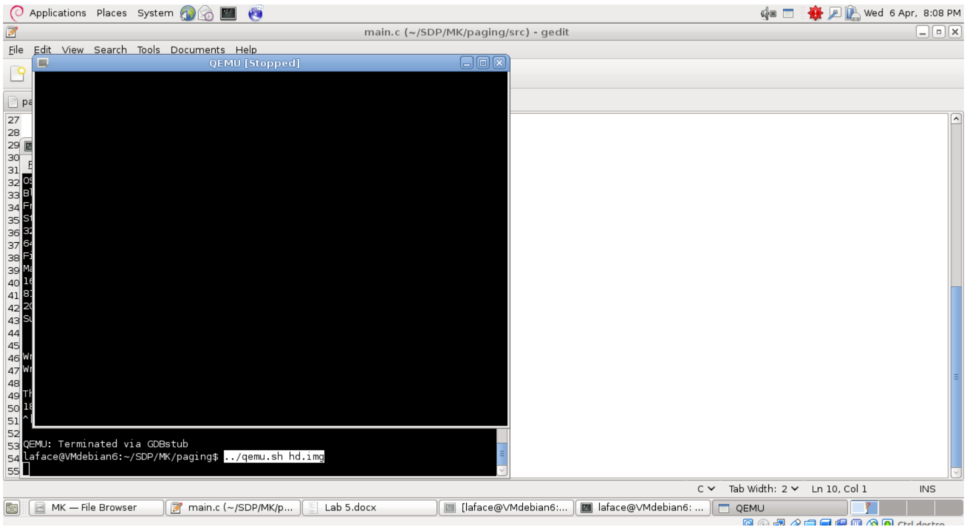
Run: **rm hd.img** to remove the image in order to avoid successive errors in the boot phase.

Run: **./build.sh ; cat src/make_stderr.txt** to run the build script and to see the output of the compilation. Strange warnings can be discarded

Run: **sudo ../pack.sh 520** to create the **hd.img** file

Run qemu in debug mode

Run: **../qemu.sh hd.img** and a black window should appear (in a Stopped state)


Applications Places System
Wed 6 Apr, 8:08 PM
main.c (~/SDP/MK/paging/src) - gedit
File Edit View Search Tools Documents Help
QEMU [Stopped]
C Tab Width: 2 Ln 10, Col 1 INS
MK — File Browser
main.c (~/SDP/MK/p...
Lab 5.docx
[laface@VMdebian6:...
laface@VMdebian6: ...
QEMU

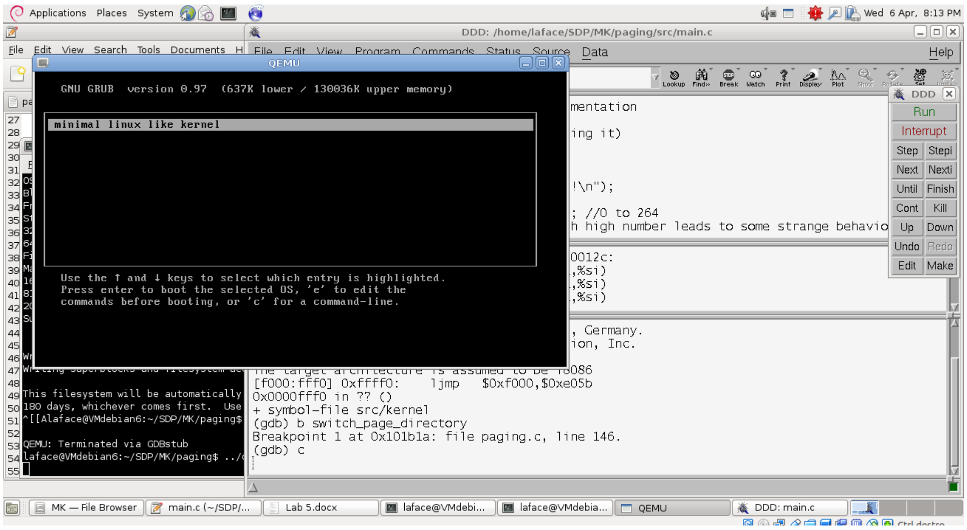
```
QEMU: Terminated via GDBstub
laface@VMdebian6:~/SDP/MK/paging$ ../qemu.sh hd.img
```

Now open another terminal in the same directory (**/home/laface/SDP/MK/paging**) and run: **ddd**

Inside the ddd command window type the following commands:

b switch_page_directory to create a breakpoint inside the function switch_page_directory.

c to continue (this should resume the execution of qemu, making it display the grub boot menu (like in the picture).

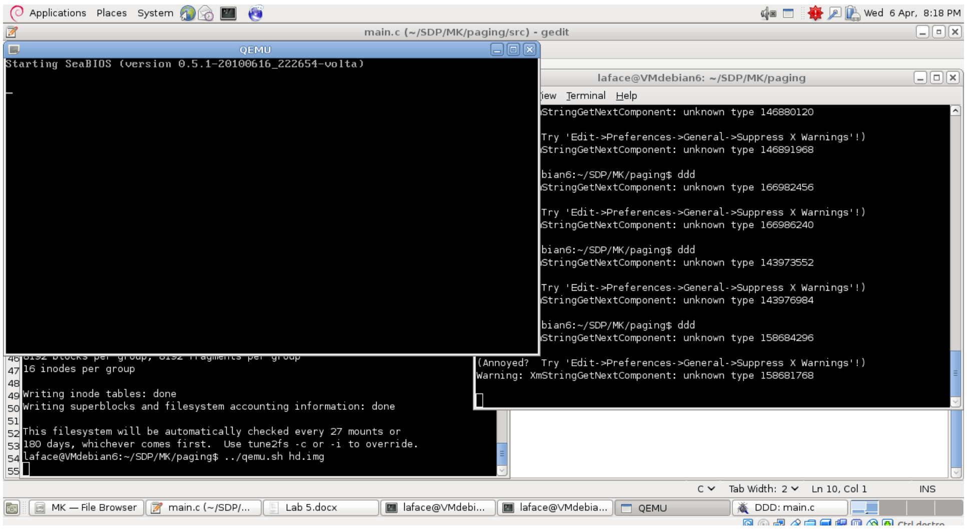


```
GNU GRUB  version 0.97  (637K lower / 130036K upper memory)

 minimal linux like kernel

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.
```

```
This filesystem will be automatically
180 days, whichever comes first.  Use
^[[Alaface@VMdebian6:~/SDP/MK/paging$

QEMU: Terminated via GDBstub
laface@VMdebian6:~/SDP/MK/paging$ ../
```

```
mentation
ing it)
!\n");
; //0 to 264
h high number leads to some strange behavio
0012c:
,%si)
,%si)
,%si)
, Germany.
ion, Inc.
The target architecture is assumed to be i8086
[f000:fff0] 0xffff0:    ljmp   $0xf000,$0xe05b
0x0000fff0 in ?? ()
+ symbol-file src/kernel
(gdb) b switch_page_directory
Breakpoint 1 at 0x101b1a: file paging.c, line 146.
(gdb) c
```

If instead you see something else instead of grub like in the following image with blinking strings, it means that you need to rebuild (remove, build and pack).

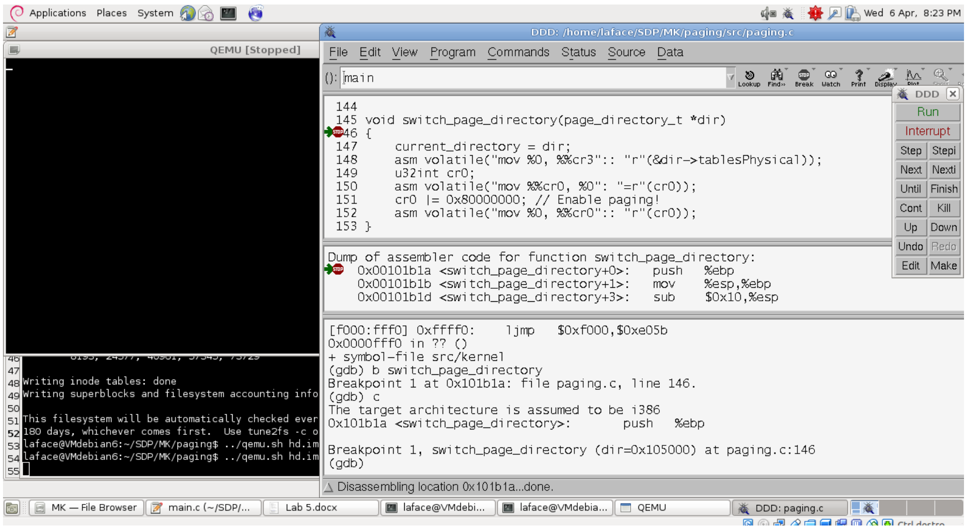


```
Starting SeaBIOS (version 0.5.1-20100616_222654-volta)
_
```

```
StringGetNextComponent: unknown type 146880120

Try 'Edit->Preferences->General->Suppress X Warnings'!)
StringGetNextComponent: unknown type 146891968

bian6:~/SDP/MK/paging$ ddd
StringGetNextComponent: unknown type 166982456

Try 'Edit->Preferences->General->Suppress X Warnings'!)
StringGetNextComponent: unknown type 166986240

bian6:~/SDP/MK/paging$ ddd
StringGetNextComponent: unknown type 143973552

Try 'Edit->Preferences->General->Suppress X Warnings'!)
StringGetNextComponent: unknown type 143976984

bian6:~/SDP/MK/paging$ ddd
StringGetNextComponent: unknown type 158684296

(Annoyed?  Try 'Edit->Preferences->General->Suppress X Warnings'!)
Warning: XmStringGetNextComponent: unknown type 158681768
```

```
8192 blocks per group, 8192 fragments per group
16 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 27 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
laface@VMdebian6:~/SDP/MK/paging$ ../qemu.sh hd.img
```

Now press enter into qemu and you should see qemu in state stopped and ddd pointing with a green array to the current istruction



```
Writing inode tables: done
Writing superblocks and filesystem accounting info

This filesystem will be automatically checked ever
180 days, whichever comes first.  Use tune2fs -c o
laface@VMdebian6:~/SDP/MK/paging$ ../qemu.sh hd.im
laface@VMdebian6:~/SDP/MK/paging$ ../qemu.sh hd.im
```

```
144
145 void switch_page_directory(page_directory_t *dir)
146 {
147     current_directory = dir;
148     asm volatile("mov %0, %%cr3":: "r"(&dir->tablesPhysical));
149     u32int cr0;
150     asm volatile("mov %%cr0, %0": "=r"(cr0));
151     cr0 |= 0x80000000; // Enable paging!
152     asm volatile("mov %0, %%cr0":: "r"(cr0));
153 }
```

```
Dump of assembler code for function switch_page_directory:
   0x00101b1a <switch_page_directory+0>:   push   %ebp
   0x00101b1b <switch_page_directory+1>:   mov    %esp,%ebp
   0x00101b1d <switch_page_directory+3>:   sub    $0x10,%esp
```

```
[f000:fff0] 0xffff0:    ljmp   $0xf000,$0xe05b
0x0000fff0 in ?? ()
+ symbol-file src/kernel
(gdb) b switch_page_directory
Breakpoint 1 at 0x101b1a: file paging.c, line 146.
(gdb) c
The target architecture is assumed to be i386
0x101b1a <switch_page_directory>:       push   %ebp

Breakpoint 1, switch_page_directory (dir=0x105000) at paging.c:146
(gdb)
```

Disassembling location 0x101b1a...done.

Now you can debug.

Exercise 1

Only the main.c needs to be modified.

Create a loop that tries to access at increasing addresses (0x0, 0x1000, 0x2000, …) till a page fault occurrs.

```
u32int i = 0;
u32int *ptr = 0;
u32int do_page_fault;
while(1) {
  do_page_fault = *ptr;
  monitor_write("Normal access at address ");
  monitor_write_hex(ptr);
  monitor_write(" at page ");
  monitor_write_dec(i);
  monitor_write("\n");
  ptr = ptr + (u32int)0x400;
  i++;
}
```

Exercise 2

To swap the mapping between the pages and frames, you need to:

1 – Go back to real mode (without the paging mechanism)

2 – modify the informations on the pages

3 – turn on again the paging mechanism

To be able to perform step 1 and three, be inspired by the switch_page_directory function: after setting up sr3, it reads the content of sr0 and places its value inside the variable sr0; then the 31th bit is turned to the value 1 by doing a OR with the appropriate mask (to leave the other bits as they are); at the end the value of the variable is loaded into the register.

Instead to be able to perform the modifications in step 2, you need to modify the information inside the global variable kernel_directory inside the paging.c file.

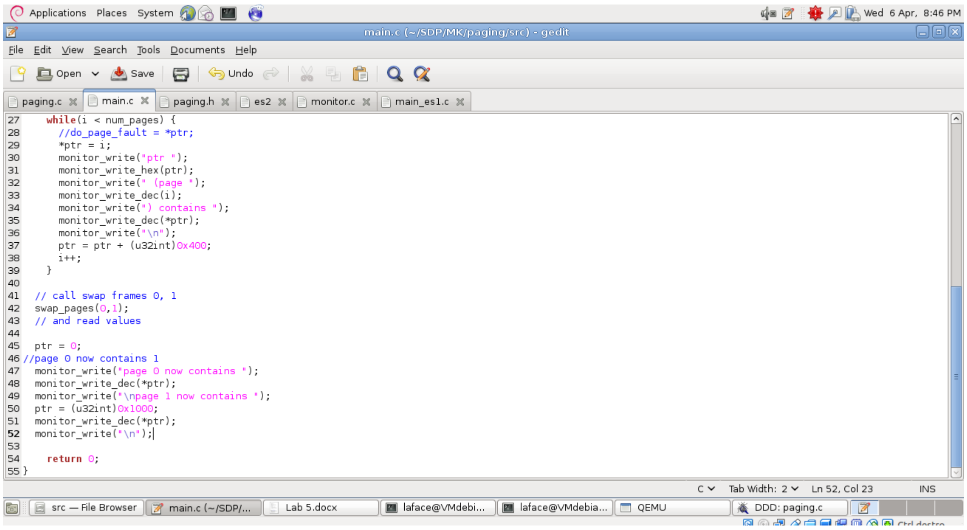
I created inside the file paging.c the following function:

```
void swap_pages(u32int pn1, u32int pn2) {
  u32int fr_n1, fr_n2;

  page_directory_t *dir = kernel_directory;
  //disable paging
  u32int cr0;
 asm volatile("mov %%cr0, %0": "=r"(cr0)); // read register cr0
 cr0 ^= 0x80000000; // Disable paging by turning off the 31th bit (^ is a XOR)
 asm volatile("mov %0, %%cr0":: "r"(cr0)); // write register cr0

  // swap pages
  fr_n1 = dir->tables[0]->pages[pn1].frame;
  fr_n2 = dir->tables[0]->pages[pn2].frame;
  dir->tables[0]->pages[pn1].frame = fr_n2;
  dir->tables[0]->pages[pn2].frame = fr_n1;

  // now re-enable paging
  asm volatile("mov %%cr0, %0": "=r"(cr0)); // read register cr0
 cr0 |= 0x80000000; // Enable paging by turning on the 31th bit
 asm volatile("mov %0, %%cr0":: "r"(cr0)); // write register cr0
}
```

Then in paging.h i make the function visible declaring the prototype:

```
void swap_pages(u32int pn1, u32int pn2);
```

And I use it in the main:


Applications Places System
Wed 6 Apr, 8:46 PM
main.c (~/SDP/MK/paging/src) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
paging.c main.c paging.h es2 monitor.c main_es1.c
C Tab Width: 2 Ln 52, Col 23 INS
src — File Browser main.c (~/SDP/... Lab 5.docx laface@VMdebi... laface@VMdebia... QEMU DDD: paging.c

```
    while(i < num_pages) {
      //do_page_fault = *ptr;
      *ptr = i;
      monitor_write("ptr ");
      monitor_write_hex(ptr);
      monitor_write(" (page ");
      monitor_write_dec(i);
      monitor_write(") contains ");
      monitor_write_dec(*ptr);
      monitor_write("\n");
      ptr = ptr + (u32int)0x400;
      i++;
    }

  // call swap frames 0, 1
  swap_pages(0,1);
  // and read values

  ptr = 0;
//page 0 now contains 1
  monitor_write("page 0 now contains ");
  monitor_write_dec(*ptr);
  monitor_write("\npage 1 now contains ");
  ptr = (u32int)0x1000;
  monitor_write_dec(*ptr);
  monitor_write("\n");

    return 0;
}
```

If strange behavior is detected, try to set the value of num_pages less than the number found in exercise 1. Writing thing to pages >= 160 makes strange things happen. The professor himself in his code limits to 160 pages (0 to 159).