HO CHI MINH UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

DIGITAL SIGNAL PROCESSING

STUDENT REPORT

Student's name	: Nguyen Duy Tinh
Student's ID	: 1852797
Lecturer	: PhD. Dinh Quang Thinh

October 26, 2020

1 Adding new system call

1.1 Introduction of System Call

A system call is a way for programs to interact with the operating system. A computer program makes a system call when it makes a request to the operating system's kernel. System calls are used for hardware services, to create or execute a process, and for communicating with kernel services, including application and process scheduling.

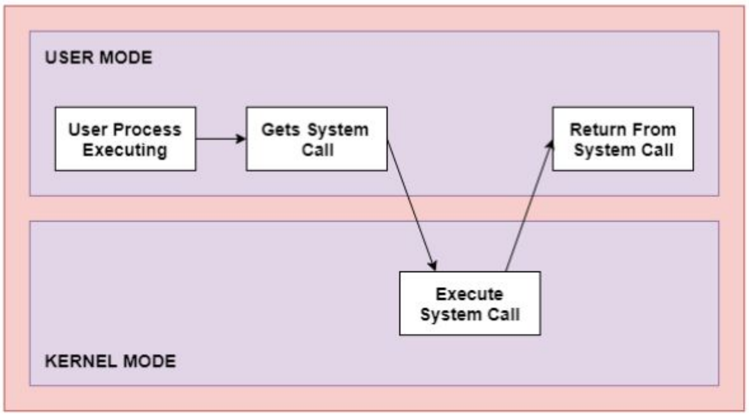


Figure 1.1

As can be seen from this diagram, the processes execute normally in the user mode until a system call interrupts this. Then the system call is executed on a priority basis in the kernel mode. After the execution of the system call, the control returns to the user mode and execution of user processes can be resumed.

In general, system calls are required in the following situations:

- If a file system requires the creation or deletion of files. Reading and writing from files also require a system call.
- Creation and management of new processes.
- Network connections also require system calls. This includes sending and receiving packets.
- Network connections also require system calls. This includes sending and receiving packets.
- Access to hardware devices (e.g. a printer, scanner) requires a system call.

Besides, system calls are mostly accessed by programs via a higher-level Application Program Interface (API) rather than direct system call use.

1.2 Objective of report

The big exercise requires that we install the kernel according to the specified version with the knowledge we learned about memory organization before setting up the file "procmem.c" as required. Then, build and compile two files "procmem.h" and "procmem.c" to display the following parameters:

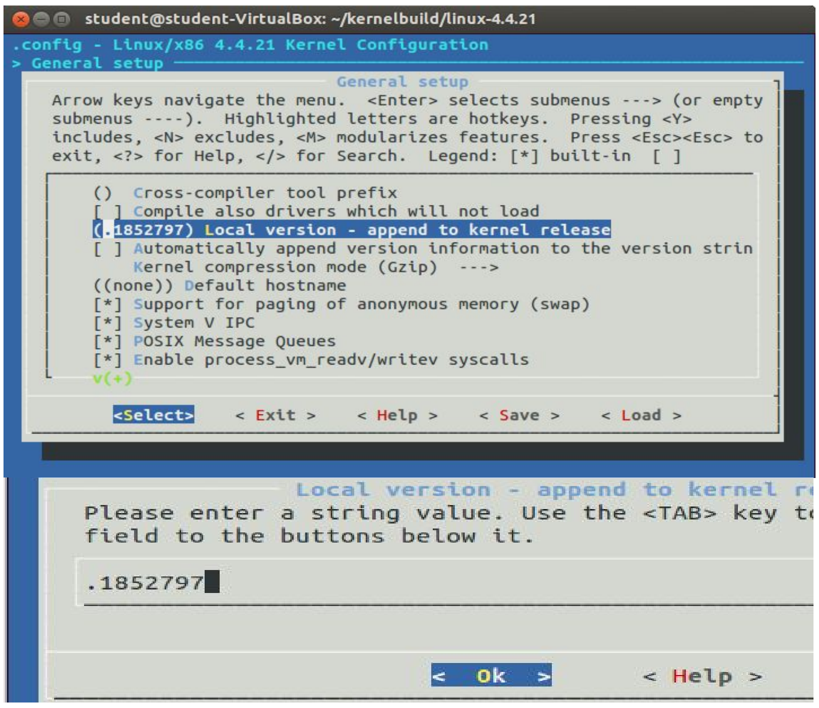
- PID.
- Student ID.
- Code segment.
- Data segment.
- Heap segment.
- Start stack.

Question: Why do we need to install kernel-package?

Answer:

We need to install kernel-package because we have a lot of kernel versions to choose from in this package, which allows us to choose according to criteria (i.e. Student ID) that match the hardware configuration of our virtual machine or real machine as well.

After executing the install and copy commands, we proceed to change the local version of the kernel to the student number.



```
student@student-VirtualBox: ~/kernelbuild/linux-4.4.21
.config - Linux/x86 4.4.21 Kernel Configuration
> General setup
                              General setup
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

      ()  Cross-compiler tool prefix
      [ ] Compile also drivers which will not load
      (.1852797) Local version - append to kernel release
      [ ] Automatically append version information to the version strin
          Kernel compression mode (Gzip)  --->
      ((none)) Default hostname
      [*] Support for paging of anonymous memory (swap)
      [*] System V IPC
      [*] POSIX Message Queues
      [*] Enable process_vm_readv/writev syscalls
      v(+)

        <Select>    < Exit >    < Help >    < Save >    < Load >

                  Local version - append to kernel r
      Please enter a string value. Use the <TAB> key t
      field to the buttons below it.

      .1852797

                                     <  Ok  >        < Help >
```

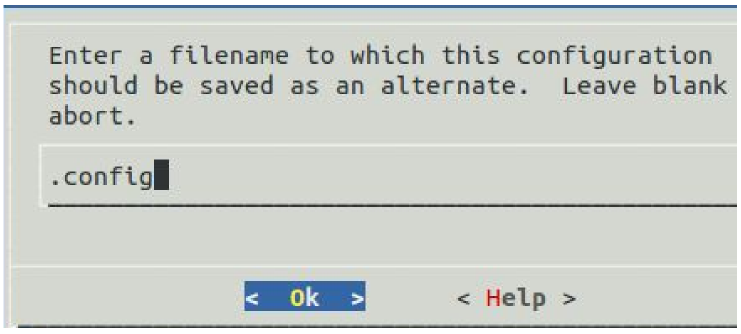
Figure 1.2

Question: Why do we have to use another kernel source from the server such as http://www.kernel.org, can we compile the original kernel (the local kernel on the running OS) directly?

Answer:

Installing lower kernel versions will help minimize system mismatches including handling special hardware needs, or hardware conflicts with pre-supplied kernels and optimize the kernel by removing useless drivers to speed up booting. However, **we still can compile the original kernel**; the kernel cannot be compiled without a compiler, but it can be installed from a compiled binary. Usually, when you install an operating system, you install an pre-compiled kernel, which was compiled by someone else. Moreover, only if you want to compile the kernel yourself, you need the source and the compiler, and all the other tools.

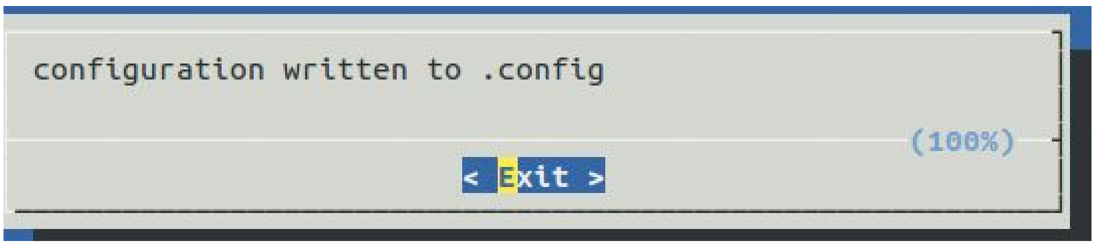
As shown in the picture, the student ID will be saved in .config in the kernel dictionary.



```
Enter a filename to which this configuration
should be saved as an alternate.  Leave blank
abort.

.config
```

Figure 1.3



```
configuration written to .config
```

Figure 1.4

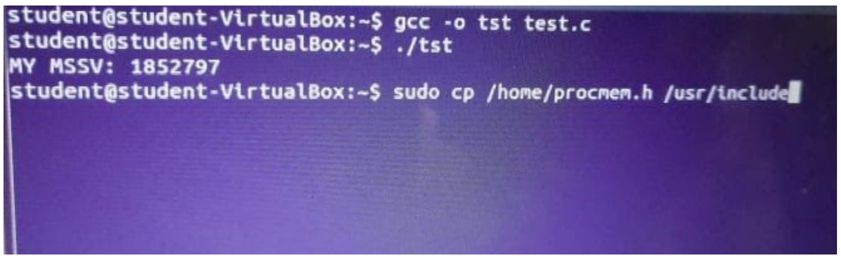
2 System call Implementation

Question: What is the meaning of other parts, i.e. i386, procmem, and sys procmem?

Answer:

i386 is ABI (i.e. Application Binary Interface), which is the interface between two module programs, one of which is usually the library or the operating system, at the common machine code level is x64, x32, i386. Besides, **procmem** is syscall's name. Finally, **sys_procmem** is an entry point or access point, which is the name of the function to call to handle syscall. The naming convention for this function is the name of syscall prefixed with "sys_".

In this step, we set the file sys_procmem.c to arch / x86 / kernel and add a line to Makefile "obj-y + = sys_procmem.o". Then we use the provided test.c file to try and the results are as below:



```
student@student-VirtualBox:~$ gcc -o tst test.c
student@student-VirtualBox:~$ ./tst
MY MSSV: 1852797
student@student-VirtualBox:~$ sudo cp /home/procmem.h /usr/include
```

Figure 2.1

And of course the two files syscall_32.tbl and syscall_64.tbl are added two lines as suggested from the article. However, in the syscalls.h file located at "include / linux /", be careful to add prototype before "endif" otherwise when "make" and "make modules" are executed, it will return a long value. long int is not expected.

Question:

```
struct proc_segs;
asmlinkage long sys_procmem( int pid, struct proc_segs * info);
```

What is the meaning of each line above?

Answer:

Firstly, **struct proc_segs** means struct proc_segs argument of type struct. Besides, the **asmlinkage long sys_procmem (int pid, struct proc_segs * info)** is a long syscall handler function; asmlinkage is a defined tag (#define) with some gcc compiler telling the compiler that the function does not expect to find all arguments in registers optimally, but only on the CPU stack. The systemcall takes the first number argument, and allows four arguments to be passed to the real system, which are on the stack. Because all systemcall are marked by asmlinkage tag, they find the stack for arguments. It is also used to allow calling a function from assembly files.

3 Compilation and Installation process

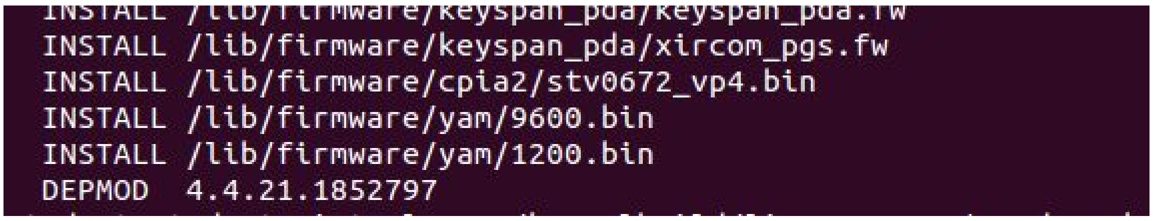
Question: What is the meaning of these two stages, namely "make" and "make modules"?

Answer:

'make' will compile the kernel image and the modules according to the configuration and 'make' compiles and links the kernel image into a single file named vmlinuz.

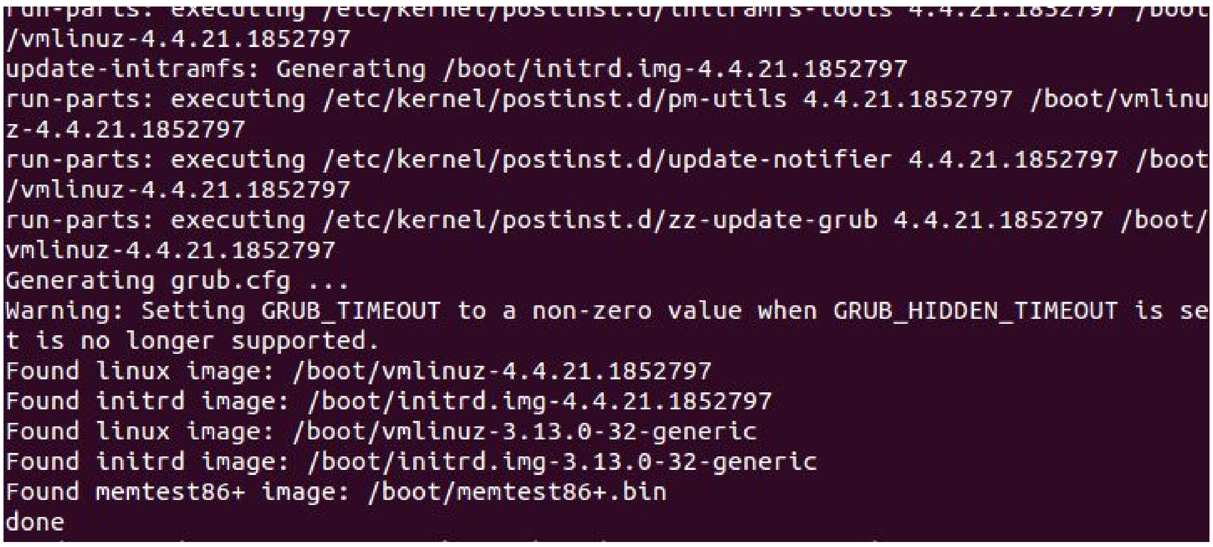
By contrast, 'make modules' will only compile the modules selected in the configuration.

When executing the "make" and "make modules" functions, be aware of the number of kernel that shows up after the execution is completed. If there is the student number present at the end then the likelihood of success is quite high and vice versa. Of course, sometimes there will be some syntax errors coming from sys_procmem.c and fixing it takes time, so you need to pay attention when coding:



```
  INSTALL /lib/firmware/keyspan_pda/keyspan_pda.fw
  INSTALL /lib/firmware/keyspan_pda/xircom_pgs.fw
  INSTALL /lib/firmware/cpia2/stv0672_vp4.bin
  INSTALL /lib/firmware/yam/9600.bin
  INSTALL /lib/firmware/yam/1200.bin
  DEPMOD  4.4.21.1852797
```

Figure 3.1



```
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 4.4.21.1852797 /boot
/vmlinuz-4.4.21.1852797
update-initramfs: Generating /boot/initrd.img-4.4.21.1852797
run-parts: executing /etc/kernel/postinst.d/pm-utils 4.4.21.1852797 /boot/vmlinu
z-4.4.21.1852797
run-parts: executing /etc/kernel/postinst.d/update-notifier 4.4.21.1852797 /boot
/vmlinuz-4.4.21.1852797
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 4.4.21.1852797 /boot/
vmlinuz-4.4.21.1852797
Generating grub.cfg ...
Warning: Setting GRUB_TIMEOUT to a non-zero value when GRUB_HIDDEN_TIMEOUT is se
t is no longer supported.
Found linux image: /boot/vmlinuz-4.4.21.1852797
Found initrd image: /boot/initrd.img-4.4.21.1852797
Found linux image: /boot/vmlinuz-3.13.0-32-generic
Found initrd image: /boot/initrd.img-3.13.0-32-generic
Found memtest86+ image: /boot/memtest86+.bin
done
```

Figure 3.2

Question: Why could this program indicate whether our system works or not?

```
#include <sys/syscall.h>
#include <stdio.h>
#define SIZE 10
int main() {
long sysvalue;
unsigned long info[SIZE];
sysvalue = syscall([number_32], 1, info);
printf("My MSSV: %ul\n", info[0]);
}
```

Answer:

This program will always execute because it does not contain any errors inside it and in both true or false cases syscall () will still return 0 or -1 corresponding values to ensure the program still executable.

4 Making API for system call

Question: Why do we have to re-define proc segs struct while we have already defined it inside the kernel?

Answer:

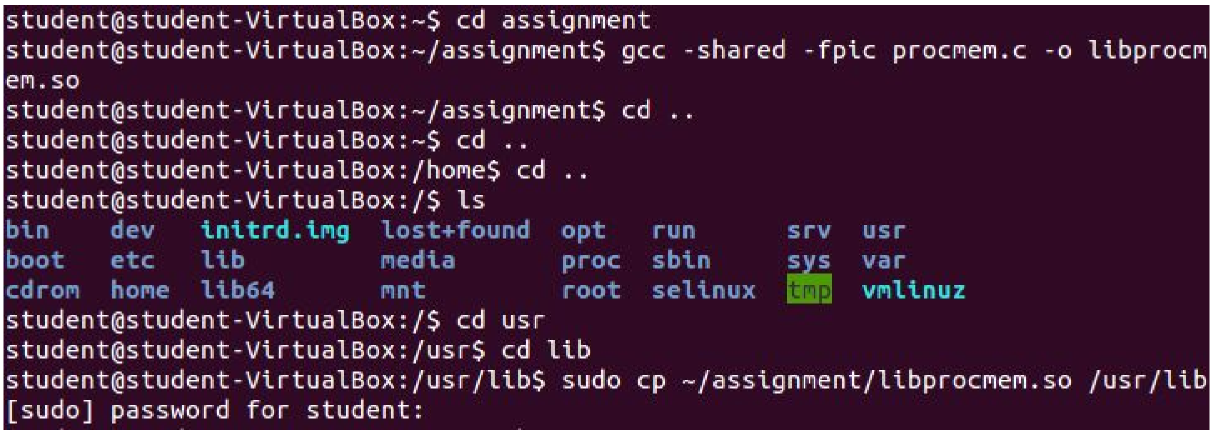
This is not redefine but define for another program. These two programs operate completely independently of each other, so there is no way they can be defined in the same place.

Question: Why root privilege (e.g. adding sudo before the cp command) is required to copy the header file to /usr/include?

Answer:

Because sudo means "super user do", accessing the space beyond /home/ is just for the owner of the operating system and /usr/include is one of that. Accessing to /usr/include is for a more powerful entity than normal users.

To make sure everyone can access to "procmem.c", we make "procmem.h" visible to GCC and share objects to allow user to integrate our system call to other application.



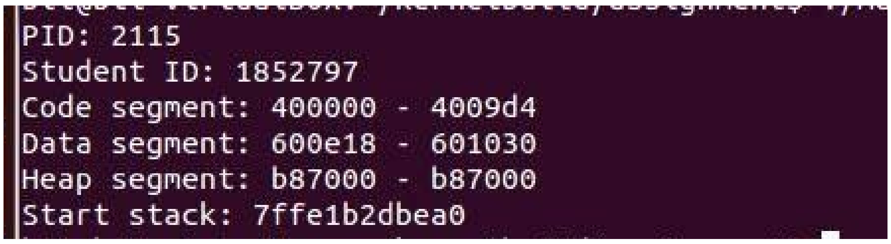
```
student@student-VirtualBox:~$ cd assignment
student@student-VirtualBox:~/assignment$ gcc -shared -fpic procmem.c -o libprocm
em.so
student@student-VirtualBox:~/assignment$ cd ..
student@student-VirtualBox:~$ cd ..
student@student-VirtualBox:/home$ cd ..
student@student-VirtualBox:/$ ls
bin     dev   initrd.img  lost+found  opt   run      srv  usr
boot    etc   lib         media       proc  sbin     sys  var
cdrom   home  lib64       mnt         root  selinux  tmp  vmlinuz
student@student-VirtualBox:/$ cd usr
student@student-VirtualBox:/usr$ cd lib
student@student-VirtualBox:/usr/lib$ sudo cp ~/assignment/libprocmem.so /usr/lib
[sudo] password for student:
```

Figure 4.1

Question: Why must we put -share and -fpic option into gcc command?

Answer:

They are used to share objects to allow users to integrate our system call to other applications.



```
PID: 2115
Student ID: 1852797
Code segment: 400000 - 4009d4
Data segment: 600e18 - 601030
Heap segment: b87000 - b87000
Start stack: 7ffe1b2dbea0
```

Figure 4.2

And finally, after a lot of time building and compiling, the main method finally got the expected results. The results include the PID, the student ID, the start and end segment of Code, Data, and the Heap, as well as the start and end index of the stack.

THE END

Thank you for your time.