SISTEMAS OPERACIONAIS

Edwin Luis Choquehuanca Mamani

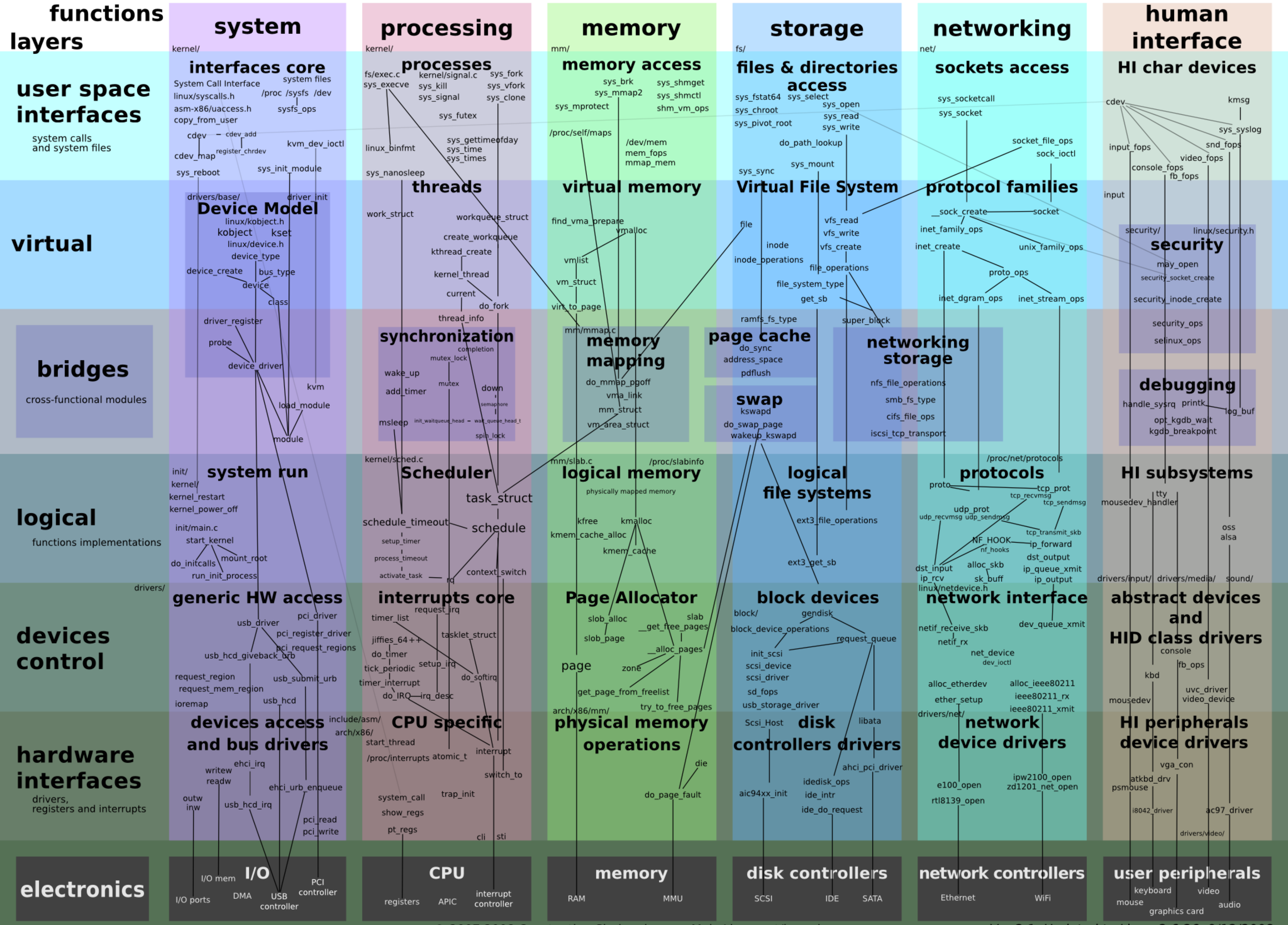
Roteiro

- O kernel de Linux
- System Calls
  - Criando uma System Call
- Compilação do Kernel (Ubuntu)
- Instalação do Kernel (Ubuntu)
- Execução da System Call

O Kernel de Linux

- Os principais módulos do kernel são:
- System (kernel/)
- Memory (mm/)
- Processing (kernel/)
- Storage (fs/)
- Networking (net/)

Linux kernel map



functions
layers

system
kernel/

processing
kernel/

memory
mm/

storage
fs/

networking
net/

human interface

user space interfaces
system calls and system files

interfaces core
System Call Interface
system files
linux/syscalls.h
/proc /sysfs /dev
asm-x86/uaccess.h
sysfs_ops
copy_from_user
cdev
cdev_add
register_chrdev
kvm_dev_ioctl
cdev_map
sys_reboot
sys_init_module

processes
fs/exec.c
sys_execve
kernel/signal.c
sys_kill
sys_signal
sys_fork
sys_vfork
sys_clone
sys_futex
linux_binfmt
sys_gettimeofday
sys_time
sys_times
sys_nanosleep

memory access
sys_brk
sys_mmap2
sys_shmget
sys_shmctl
shm_vm_ops
sys_mprotect
/proc/self/maps
/dev/mem
mem_fops
mmap_mem

files & directories access
sys_fstat64
sys_select
sys_chroot
sys_open
sys_read
sys_pivot_root
sys_write
do_path_lookup
sys_mount
sys_sync

sockets access
sys_socketcall
sys_socket
socket_file_ops
sock_ioctl

HI char devices
cdev
kmsg
sys_syslog
input_fops
snd_fops
video_fops
console_fops
fb_fops

virtual

drivers/base/
driver_init
Device Model
linux/kobject.h
kobject
kset
linux/device.h
device_type
device_create
bus_type
device
class

threads
work_struct
workqueue_struct
create_workqueue
kthread_create
kernel_thread
current
do_fork
thread_info

virtual memory
find_vma_prepare
vmalloc
vmlist
vm_struct
virt_to_page

Virtual File System
file
vfs_read
vfs_write
inode
vfs_create
inode_operations
file_operations
file_system_type
get_sb
super_block

protocol families
__sock_create
socket
inet_family_ops
inet_create
unix_family_ops
proto_ops
inet_dgram_ops
inet_stream_ops

input
security/
linux/security.h
security
may_open
security_socket_create
security_inode_create
security_ops
selinux_ops

bridges
cross-functional modules

driver_register
probe
device_driver
kvm
load_module
module

synchronization
completion
mutex_lock
wake_up
mutex
add_timer
down
semaphore
msleep
init_waitqueue_head
wait_queue_head_t
spin_lock

mm/mmap.c
memory mapping
do_mmap_pgoff
vma_link
mm_struct
vm_area_struct

ramfs_fs_type
page cache
do_sync
address_space
pdflush

swap
kswapd
do_swap_page
wakeup_kswapd

networking storage
nfs_file_operations
smb_fs_type
cifs_file_operations
iscsi_tcp_transport

debugging
handle_sysrq
printk
log_buf
opt_kgdb_wait
kgdb_breakpoint

logical
functions implementations

system run
init/
kernel/
kernel_restart
kernel_power_off
init/main.c
start_kernel
mount_root
do_initcalls
run_init_process

kernel/sched.c
Scheduler
task_struct
schedule_timeout
schedule
setup_timer
process_timeout
activate_task
rq
context_switch

mm/slab.c
/proc/slabinfo
logical memory
physically mapped memory
kfree
kmalloc
kmem_cache_alloc
kmem_cache

logical file systems
ext3_file_operations
ext3_get_sb

/proc/net/protocols
protocols
proto
tcp_prot
tcp_recvmsg
tcp_sendmsg
udp_prot
udp_recvmsg
udp_sendmsg
tcp_transmit_skb
NF_HOOK
nf_hooks
ip_forward
dst_output
dst_input
alloc_skb
ip_queue_xmit
ip_rcv
sk_buff
ip_output

HI subsystems
tty
mousedev_handler
oss
alsa

devices control

drivers/
drivers/input/
drivers/media/
sound/

generic HW access
pci_driver
usb_driver
pci_register_driver
pci_request_regions
usb_hcd_giveback_urb
request_region
usb_submit_urb
request_mem_region
usb_hcd
ioremap

interrupts core
request_irq
timer_list
tasklet_struct
jiffies_64++
do_timer
tick_periodic
setup_irq
do_softirq
timer_interrupt
do_IRQ
irq_desc

Page Allocator
slob_alloc
slab
__get_free_pages
slob_page
__alloc_pages
page
zone
get_page_from_freelist
try_to_free_pages

block devices
block/
gendisk
block_device_operations
request_queue
init_scsi
scsi_device
scsi_driver
sd_fops
usb_storage_driver

linux/netdevice.h
network interface
netif_receive_skb
dev_queue_xmit
netif_rx
net_device
dev_ioctl
alloc_etherdev
alloc_ieee80211
ether_setup
ieee80211_rx
ieee80211_xmit

abstract devices and HID class drivers
console
fb_ops
kbd
uvc_driver
video_device
mousedev

hardware interfaces
drivers, registers and interrupts

devices access and bus drivers
include/asm/
arch/x86/
ehci_irq
writew
readw
ehci_urb_enqueue
outw
inw
usb_hcd_irq
pci_read
pci_write

CPU specific
start_thread
/proc/interrupts
atomic_t
interrupt
switch_to
system_call
trap_init
show_regs
pt_regs
cli
sti

arch/x86/mm/
physical memory operations
die
do_page_fault

disk controllers drivers
Scsi_Host
libata
ahci_pci_driver
idedisk_ops
aic94xx_init
ide_intr
ide_do_request

drivers/net/
network device drivers
e100_open
ipw2100_open
zd1201_net_open
rtl8139_open

HI peripherals device drivers
vga_con
atkbd_drv
psmouse
i8042_driver
ac97_driver
drivers/video/

electronics

I/O
I/O mem
I/O ports
DMA
USB controller
PCI controller

CPU
registers
APIC
interrupt controller

memory
RAM
MMU

disk controllers
SCSI
IDE
SATA

network controllers
Ethernet
WiFi

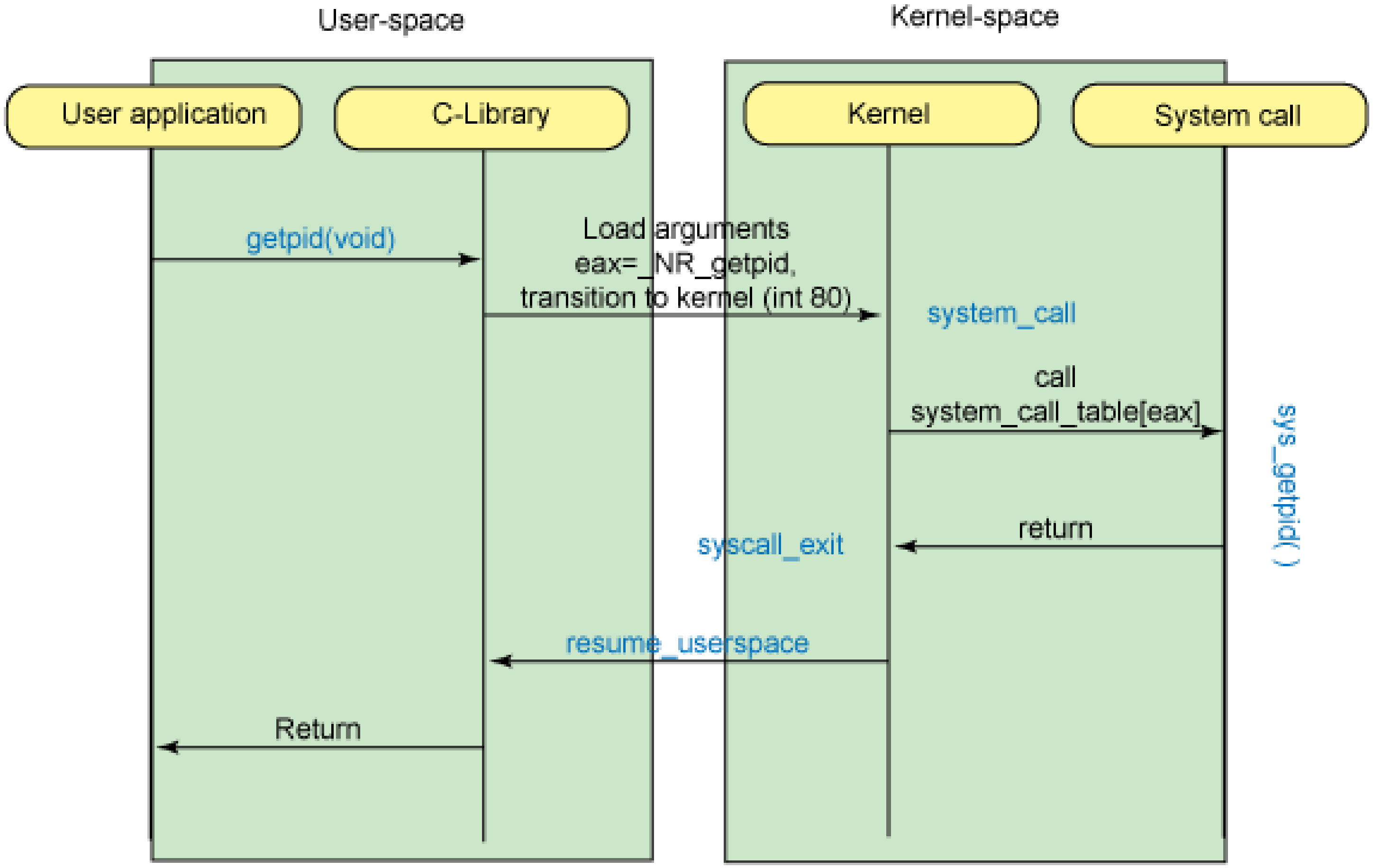
user peripherals
keyboard
mouse
video
graphics card
audio

Ver 2.1, Updated to Linux 2.6.26, 9/12/2008

System Calls

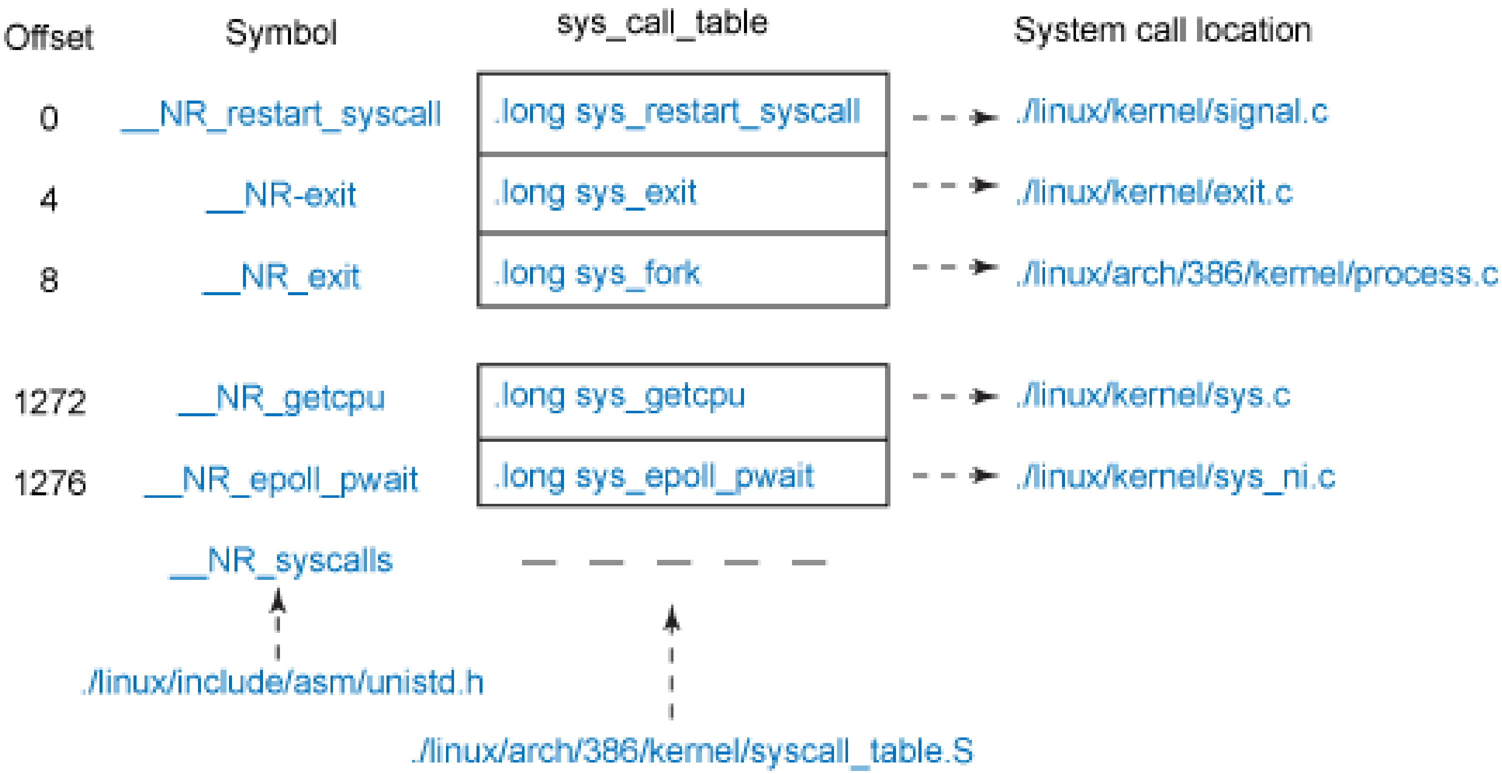
- É o modo padrão para exportar um serviço do SO para o espaço do usuário.
- As chamadas direitas não podem ser feitas.

System Calls



Fluxo da system call getpid

System Calls



Offset	Symbol	sys_call_table	System call location
0	__NR_restart_syscall	.long sys_restart_syscall	./linux/kernel/signal.c
4	__NR-exit	.long sys_exit	./linux/kernel/exit.c
8	__NR_exit	.long sys_fork	./linux/arch/386/kernel/process.c
1272	__NR_getcpu	.long sys_getcpu	./linux/kernel/sys.c
1276	__NR_epoll_pwait	.long sys_epoll_pwait	./linux/kernel/sys_ni.c
	__NR_syscalls		

./linux/include/asm/unistd.h

./linux/arch/386/kernel/syscall_table.S

Tabela de system calls

Criando uma System Call

- Config
  - arch/x86/kernel/entry_32.S
    - `#include "syscall_table_32.S" /*no final*/`
  - arch/x86/kernel/syscall_table_32.S
    - `.long sys_myservice /* 333 */`
  - arch/x86/include/unistd_32.h
    - `#define __NR_myservice 333`
- Novos arquivos
  - kernel/myservice.c

Criando uma System Call

- Novos arquivos
  - kernel/myservice.c

```
#include <linux/myservice.h>
#include <linux/kernel.h>

  asmlinkage void sys_myservice(void){
     printk(KERN_DEBUG "Ola :D ");
  }
```

Criando uma System Call

- Novos arquivos
  - include/linux/myservice.h

```
#ifndef __LINUX_MYSERVICE_H
#define __LINUX_MYSERVICE_H

#include <linux/unistd.h>
#include <linux/linkage.h>

#endif
```

Criando uma System Call

- Inserir o nome do system call no arquivo
  - kernel/Makefile

```
obj-y     = sched.o
  ...
  sched_clock.o myservice.o
```

Compilação do Kernel (Ubuntu)

- Dependências:
  - sudo apt-get update
  - sudo apt-get install build-essential kernel-package xmlto linux-source libncurses-dev
  - cd /usr/src
  - tar –xvjf linux-source-XXXX.tar.bz2
  - cp /boot/config-XXXX /usr/src/linux-source-XXXX/.config
  - make oldconfig or make menuconfig

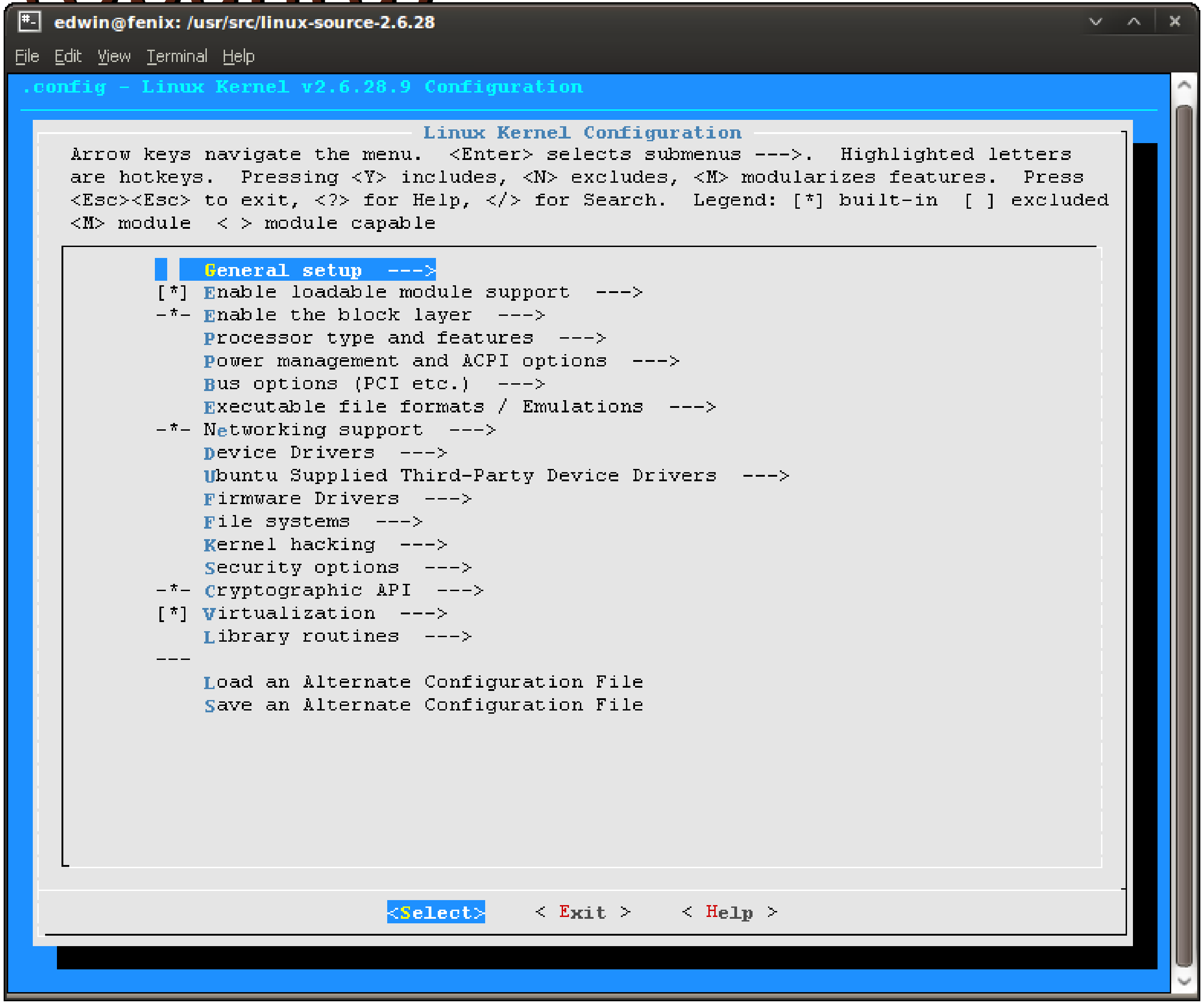
Compilação do Kernel (Ubuntu)

- Dependências:

  -
    ```
    sudo make-kpkg clean
    ```

  -
    ```
    sudo make-kpkg --append-to-version=-mysyscall --initrd kernel_image kernel_headers
    ```

Compilação do Kernel (Ubuntu)



edwin@fenix: /usr/src/linux-source-2.6.28
File Edit View Terminal Help

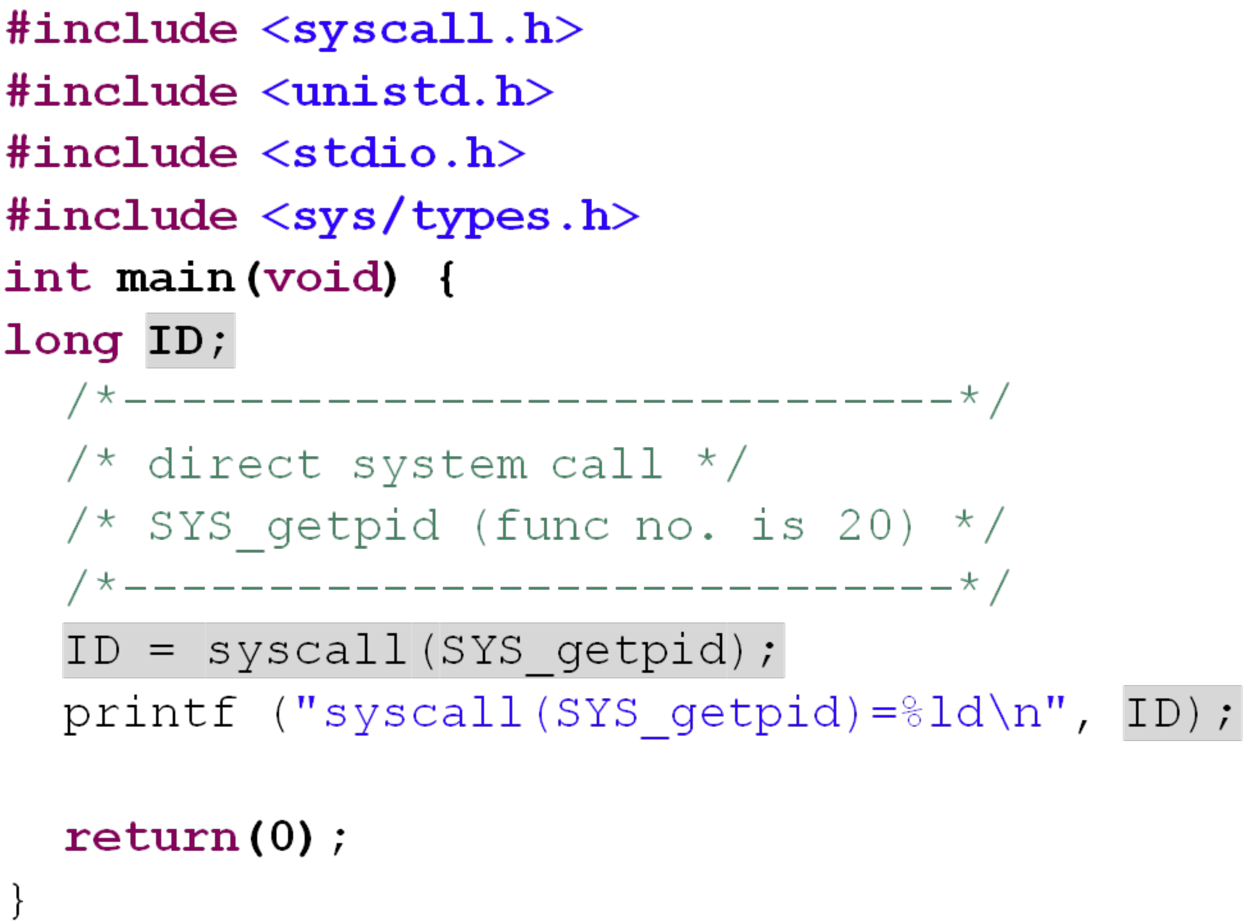
```
.config - Linux Kernel v2.6.28.9 Configuration

                         Linux Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects 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  [ ] excluded
  <M> module  < > module capable

            General setup  --->
        [*] Enable loadable module support  --->
        -*- Enable the block layer  --->
            Processor type and features  --->
            Power management and ACPI options  --->
            Bus options (PCI etc.)  --->
            Executable file formats / Emulations  --->
        -*- Networking support  --->
            Device Drivers  --->
            Ubuntu Supplied Third-Party Device Drivers  --->
            Firmware Drivers  --->
            File systems  --->
            Kernel hacking  --->
            Security options  --->
        -*- Cryptographic API  --->
        [*] Virtualization  --->
            Library routines  --->
        ---
            Load an Alternate Configuration File
            Save an Alternate Configuration File

                      <Select>    < Exit >    < Help >
```

Instalação do Kernel (Ubuntu)

- Comandos:

```
cd ..
dpkg -i linux-image-XXXX.deb
dpkg -i linux-headers-XXXX.deb
sudo reboot
```

Execução da System Call



```
#include <syscall.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
int main(void) {
long ID;
  /*----------------------------*/
  /* direct system call */
  /* SYS_getpid (func no. is 20) */
  /*----------------------------*/
  ID = syscall(SYS_getpid);
  printf ("syscall(SYS_getpid)=%ld\n", ID);

  return(0);
}
```

Execução da System Call

- Guardar o código
- Compilar
  - gcc arquivo.c –o algo
- Executar
  - ./algo

Referencias

- http://www.ibm.com/developerworks/li
- http://www.csee.umbc.edu/courses/un
- http://www.digilife.be/quickreferences/
- https://help.ubuntu.com/community/Ke
- http://www.makelinux.net/kernel_map