http://www.howopensource.com/2011/08/how-to-compile-and-install-linux-kernel-3-0-in-ubuntu-11-04-10-10-and-10-04/
1.download vim
2.uname -r //find own version
3.download linux-3.10.77.tar.gz
   from https://www.kernel.org/pub/linux/kernel/v3.x/

4.sudo apt-get install nautilus //為了檔案開啟無需權限
5.sudo nautilus //move file from download to /usr/src/kernels
6.sudo tar -zxv -f linux-3.10.77.tar.gz
7.下載編譯kernel的相關套件
sudo apt-get install libncurses5-dev
sudo apt-get update && sudo apt-get upgrade
8.將你以前進行過的核心功能選擇檔案也刪除掉
sudo make mrproper

9.處理核心項目與功能的選擇
sudo make menuconfig
//save to .config and press ok and then exit pressing enter

//sudo make -j 4 clean bzImage modules
//10.因為gcc編譯器似乎有問題，故改install這個
```
//$ sudo apt-get install gcc-multilib
```
11.切超級權限後
sudo -i
12.先清除暫存檔，再編譯核心與編譯模組
make -j 4 clean
make -j 4 bzImage
make -j 4 modules
13.模組是放置到 /lib/modules/$(uname -r) 目錄下的
make modules_install
查看是否裝進去了
ll /lib/modules/
14.模組安裝妥當後，開始安裝新核心安裝與多重開機選單grub，同時保留舊版的核心，並且新增新版的核心在我們的主機上面。
make install
15.grub-mkconfig -o /boot/grub/grub.cfg
16.
vim /etc/default/grub //將default改成0
grep default /boot/grub/grub.cfg

17.重開機後
cd /usr/src/kernels/linux-3.10.77/arch/x86/kernel
touch `linux_survey_TT.c`
後
2015/12/03

add a system call:
http://it.livekn.com/2013/01/kernel-system-call.html

一.Kernel部分

18.將位置切換到要編譯版本的資料夾

/usr/src/kernel/linux-3.10.77

19.在以下路徑增加函式的定義：
/usr/src/kernel/linux-3.10.77/include/linux/syscalls.h
asmlinkage int sys_linux_survey_TT(int pid,char* ma);

20.在以下路徑增加下列指令
/usr/src/kernel/linux-3.10.77/arch/x86/syscalls/syscall32.tbl

21.在以下路徑增加下列指令
/usr/src/kernel/linux-3.10.77/arch/x86/kernel/Makefile
obj-y +=linux_survey_TT.o

User part:
22.在以下路徑增加下列指令
/usr/include/asm-generic/unistd.h
#define __NR_linux_survery_TT 351
並注意
#define __NR_syscalls 352
（數字需改為增加的syscall數字+1）

23.在以下路徑增加下列指令
/usr/include/i386-linux-gnu/bits/syscall.h
#define SYS_linux_survey_TT __NR_linux_survey_TT

24.sudo apt-get update
    sudo apt-get upgrade
    reboot
25.看所有正在執行的process
    ps aux|more
26.找有linux_survey_TT名稱的process
    ps aux|grep linux_survey_TT

27.find process id and process name
http://linuxgazette.net/112/krishnakumar.html
28. 看

```
struct task_struct {
	volatile long state;	/* -1 unrunnable, 0 runnable, >0 stopped */
	struct thread_info *thread_info;
	atomic_t usage;
	...
	...
	...
	struct mm_struct *mm, *active_mm;
	...
	...
	...
	pid_t pid;
	...
	...
	...
	char comm[16];
	...
	...
};
struct mm_struct {
	struct vm_area_struct * mmap;		/* list of VMAs */
	struct rb_root mm_rb;
	struct vm_area_struct * mmap_cache;	/* last find_vma result */
	...
	...
	...
	unsigned long start_code, end_code, start_data, end_data;
	unsigned long start_brk, brk, start_stack;
	...
	...
	...
};
struct vm_area_struct {
	struct mm_struct * vm_mm;	/* The address space we belong to. */
	unsigned long vm_start;		/* Our start address within vm_mm. */
	unsigned long vm_end;		/* The first byte after our end address
					   within vm_mm. */
	....
	....
	....
	/* linked list of VM areas per task, sorted by address */
	struct vm_area_struct *vm_next;
	....
	....
}
```

29.. vim linux_survey_TT.c

```
static int pid_mem = 1;

static void print_mem(struct task_struct *task)
{
        struct mm_struct *mm;
        struct vm_area_struct *vma;
        int count = 0;
        mm = task->mm;
        printk("\nThis mm_struct has %d vmas.\n", mm->map_count);
        for (vma = mm->mmap ; vma ; vma = vma->vm_next) {
                printk ("\nVma number %d: \n", ++count);
                printk("  Starts at 0x%lx, Ends at 0x%lx\n",
                          vma->vm_start, vma->vm_end);
        }
        printk("\nCode  Segment start = 0x%lx, end = 0x%lx \n"
                 "Data  Segment start = 0x%lx, end = 0x%lx\n"
                 "Stack Segment start = 0x%lx\n",
                 mm->start_code, mm->end_code,
                 mm->start_data, mm->end_data,
                 mm->start_stack);
}

static int mm_exp_load(void){
        struct task_struct *task;
        printk("\nGot the process id to look up as %d.
\n", pid_mem);
        for_each_process(task) {
                if ( task->pid == pid_mem) {
                        printk("%s[%d]\n", task->comm, task->pid);
                        print_mem(task);
                }
        }
        return 0;
}

static void mm_exp_unload(void)
{
        printk("\nPrint segment information module exiting.\n");
}
module_init(mm_exp_load);
module_exit(mm_exp_unload);
module_param(pid_mem, int, 0);
```
30.每次更新完

```
make -j 4 clean
make -j 4
make -j 4 modules_install
make -j 4 install
```

https://github.com/tjjh89017/kernel-project/blob/master/arch/x86/kernel/project.c
https://nos-study.hackpad.com/Linux-Operating-System-Project-1-kX1akI0WNGn#:h=Paper-Work-(15-points)
https://nos-study.hackpad.com/Linux-Operating-System-Project-1-kX1akI0WNGn
http://blog.csdn.net/yrj/article/details/2508785