**Activity 12: Kernel Module**

**สมาชิก**

|  |  |
| --- | --- |
| **ชื่อ-นามสกุล** | **เลขประจำตัวนิสิต** |
| นายสิปปภาส ชวานนท์ | 6630333721 |
| นายเนติภัทร โพธิพันธ์ | 6631331621 |
| นายวรลภย์ ศรีชัยนนท์ | 6632200221 |

**Checkpoint #1**

**Create a dummy kernel module (dummy.c) and use the given Makefile to build the module. Use insmod, rmmod to install/remove module. You have to demonstrate that the module has been installed and removed (using dmesg and lsmod).**

ในขั้นตอนนี้ จะใช้ไฟล์ **dummy.c** ดังกล่าว

**// dummy.c**

**#include <linux/module.h> /\* Needed by all modules \*/**

**#include <linux/kernel.h> /\* Needed for KERN\_INFO \*/**

**MODULE\_LICENSE("GPL");**

**MODULE\_AUTHOR("KRERK PIROMSOPA, PH.D. <Krerk.P@chula.ac.th>");**

**MODULE\_DESCRIPTION("\"cpmod\" Dummy Kernel Module");**

**int init\_module(void)**

**{**

**printk(KERN\_INFO "CPMOD: init\n");**

**// non 0 - means init\_module failed / module can't be loaded**

**return 0;**

**}**

**void cleanup\_module(void)**

**{**

**printk(KERN\_INFO "CPMOD: cleanup\n");**

**}**

และ **Makefile** ดังกล่าว

**obj-m += dummy.o**

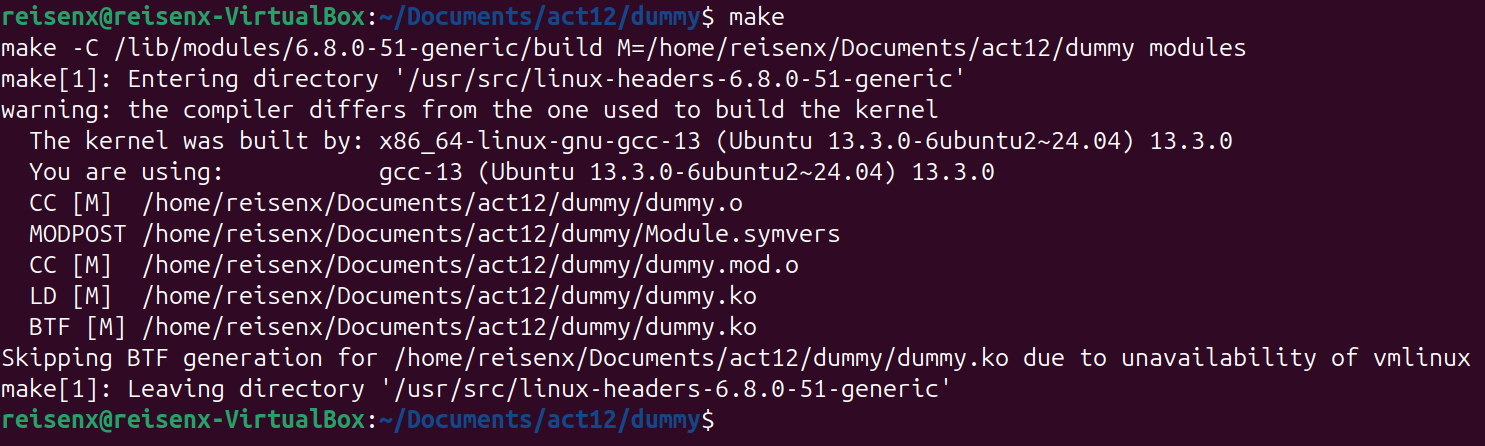
**all:**

**make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules**

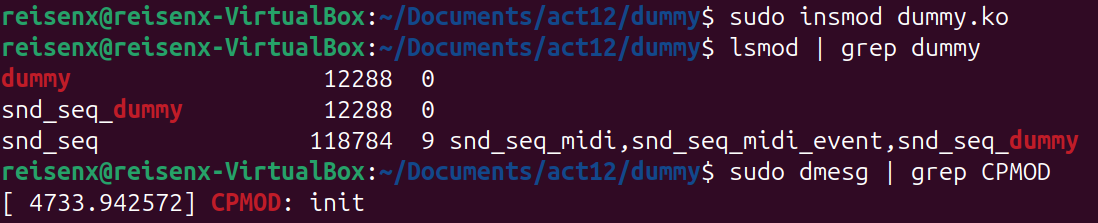
**clean:**

**make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean**

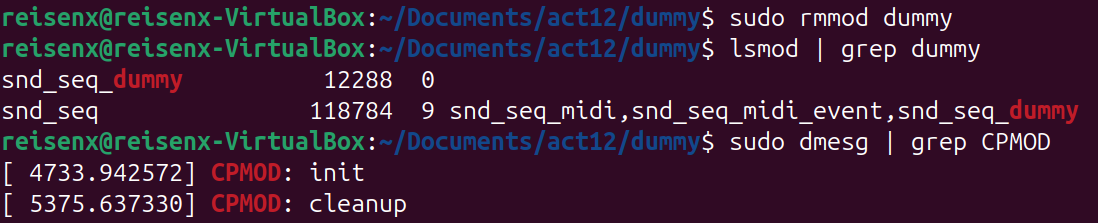
จากนั้นเปิด Ubuntu Virtual Box และใช้งานคำสั่ง **make** เพื่อทำให้ Makefile ทำงาน และ build ไฟล์ที่จำเป็นต่อการใช้งาน



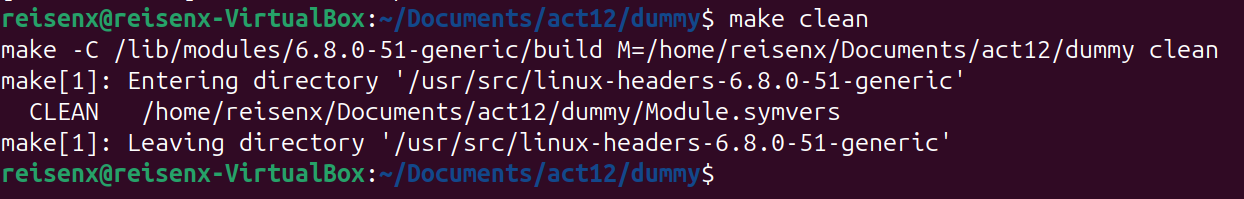
* จากนั้นใช้งานคำสั่ง **sudo insmod dummy.ko** เพื่อ load kernel module ของ dummy
* เพื่อให้มั่นใจว่า kernel dummy ของเราจะถูก load เข้าไปในระบบปฏิบัติการจริงๆ ให้ตรวจสอบโดยใช้คำสั่ง **lsmod | grep dummy** ซึ่งจะเห็นว่าในที่นี้เรามี kernel module dummy ที่ถูก load เข้าไปในระบบปฏิบัติการจริงๆ
* นอกจากนี้เรายังสามารถตรวจสอบ log messages ที่มาจาก kernel โดยการใช้คำสั่ง **sudo dmesg | grep CPMOD** เพื่อเช็คว่ามี kernel module dummy ถูก load เข้าไปในระบบปฏิบัติการ



* จากนั้นใช้งานคำสั่ง **sudo rmmod dummy** เพื่อนำ kernel module ของ dummy ออกไปจากระบบปฏิบัติการ (unload)
* เพื่อตรวจสอบว่า kernel module dummy ถูกเอาออกไปจริงๆ เราสามารถตรวจสอบ log messages โดยใช้คำสั่ง **sudo dmesg | grep CPMOD**



สุดท้ายให้ใช้งานคำสั่ง **make clean** เพื่อนำไฟล์ที่ไม่จำเป็นออกไปจากเครื่อง (clean)



**Checkpoint #2**

**In this exercise, you have to compile the given character driver**

**Use insmod to install the driver. Once the driver is installed, use the mknod command to create /dev/osinfo device file. You may obtain the major device number from dmesg. Please also show the content of /dev/osinfo.**

ในขั้นตอนนี้ จะใช้ไฟล์ **osinfo.c** ดังกล่าว

**// osinfo.c**

**#include <linux/module.h>**

**#include <linux/kernel.h>**

**#include <linux/fs.h>**

**#include <asm/uaccess.h>**

**/\* Needed by all modules \*/**

**/\* Needed for KERN\_INFO \*/**

**MODULE\_LICENSE("GPL");**

**MODULE\_AUTHOR("KRERK PIROMSOPA, PH.D. <Krerk.P@chula.ac.th>");**

**MODULE\_DESCRIPTION("\"osinfo\" Character Device");**

**#define DEVICENAME "osinfo"**

**static int dev\_major;**

**static int dev\_open = 0;**

**static char \*f\_ptr;**

**static const char f\_data0[] =**

**"0:CP ENG CU OS 2022S2 - Instructors\n"**

**"1:\tVeera Muangsin, Ph.D.\n"**

**"2:\tKrerk Piromsopa, Ph.D.\n"**

**"3:\tThongchai Rojkangsadan\n";**

**// prototypes for device functions**

**static int device\_open(struct inode \*, struct file \*);**

**static int device\_release(struct inode \*inode, struct file \*file);**

**static ssize\_t device\_read(struct file \*, char \*, size\_t, loff\_t \*);**

**// File operations structor**

**// Only implement those that will be used.**

**static struct file\_operations dev\_fops = {**

**.read    = device\_read,**

**.open    = device\_open,**

**.release = device\_release**

**};**

**int init\_module(void)**

**{**

**printk(KERN\_INFO "CPCHAR: dev osinfo init\n");**

**dev\_major = register\_chrdev(0, DEVICENAME, &dev\_fops);**

**if (dev\_major < 0) {**

**printk(KERN\_ALERT "Fail register\_chrdev osinfo with %d\n", dev\_major);**

**return dev\_major;**

**}**

**printk(KERN\_INFO "Device MajorNumber %d.\n", dev\_major);**

**printk(KERN\_INFO "To create a device file:\n");**

**printk(KERN\_INFO "\t'mknod /dev/%s c %d 0'.\n", DEVICENAME, dev\_major);**

**printk(KERN\_INFO "Try varying minor numbers.\n");**

**printk(KERN\_INFO "Please remove the device file and module when done.\n");**

**/\* \* non 0 - means init\_module failed \*/**

**return 0;**

**}**

**void cleanup\_module(void)**

**{**

**printk(KERN\_INFO "CPCHAR: dev osinfo cleanup\n");**

**unregister\_chrdev(dev\_major, DEVICENAME);**

**}**

**static int device\_open(struct inode \*inode, struct file \*file)**

**{**

**if (dev\_open)**

**return -EBUSY;**

**dev\_open++;**

**printk(KERN\_INFO "dev minor %d\n", MINOR(inode->i\_rdev));**

**f\_ptr = (char \*)f\_data0;**

**// lock module**

**try\_module\_get(THIS\_MODULE);**

**return 0;**

**}**

**static int device\_release(struct inode \*inode, struct file \*file)**

**{**

**dev\_open--; /\* We're now ready for our next caller \*/**

**// release module**

**module\_put(THIS\_MODULE);**

**return 0;**

**}**

**static ssize\_t device\_read(struct file \*filp,**

**char \*buffer,**

**/\* see include/linux/fs.h \*/**

**/\* buffer to fill with data \*/**

**/\* length of the buffer \*/**

**size\_t length,**

**loff\_t \*offset)**

**{**

**int bytes\_read = 0;**

**if (\*f\_ptr == 0) {**

**return 0;**

**}**

**while (length && \*f\_ptr) {**

**put\_user(\*(f\_ptr++), buffer++);**

**length--;**

**bytes\_read++;**

**}**

**return bytes\_read;**

**}**

และสร้าง **Makefile** โดยมีลักษณะดังนี้

**obj-m += osinfo.o**

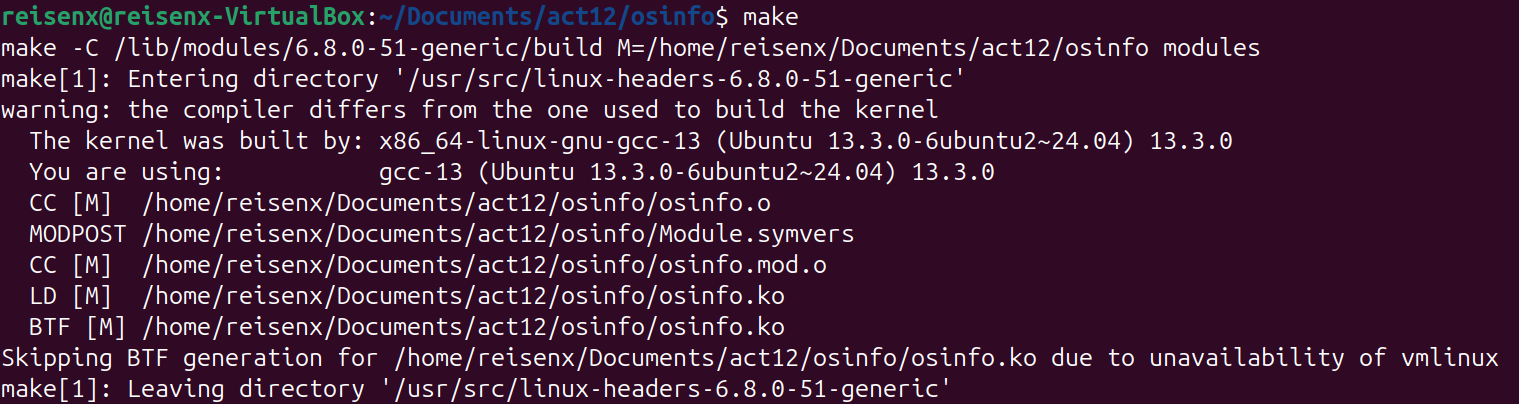
**all:**

**make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules**

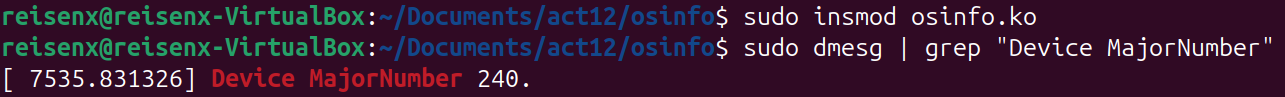
**clean:**

**make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean**

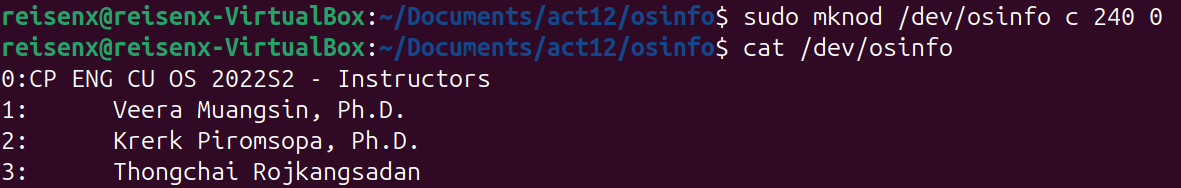
ใช้งานคำสั่ง **make** เพื่อทำให้ Makefile ทำงาน และ build ไฟล์ที่จำเป็นต่อการใช้งาน



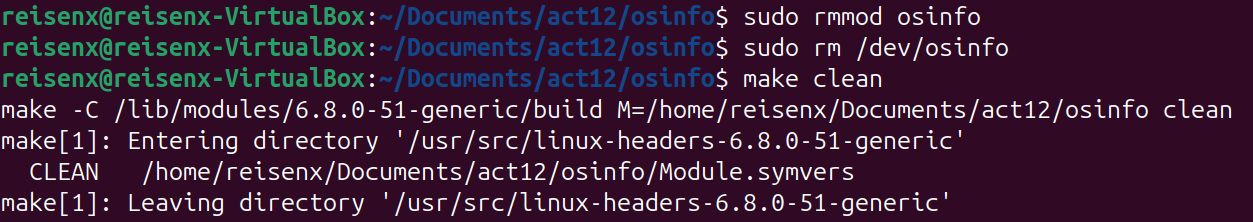
* ใช้งานคำสั่ง **sudo insmod osinfo.ko** เพื่อ load kernel module ของ osinfo
* จากนั้นตรวจสอบ MajorNumber โดยการใช้งานคำสั่ง **sudo dmesg | grep “Device MajorNumber”**



* ใช้งานคำสั่ง **sudo mknod /dev/osinfo c 240 0** เพื่อสร้าง Character Device ชื่อว่า osinfo ในโฟลเดอร์ dev
* แสดงผลไฟล์ Character Device ที่ถูกสร้างขึ้นมาด้วยคำสั่ง **cat /dev/osinfo**



* ใช้งานคำสั่ง **sudo rmmod osinfo** เพื่อนำ kernel module ของ osinfo ออกไปจากระบบปฏิบัติการ (unload)
* ลบโฟล์เดอร์และไฟล์ที่ถูกสร้างขึ้นมา (/dev/osinfo) ด้วยคำสั่ง **sudo rm /dev/osinfo**
* สุดท้ายให้ใช้งานคำสั่ง **make clean** เพื่อนำไฟล์ที่ไม่จำเป็นออกไปจากเครื่อง (clean)



**Checkpoint #3**

**Modify the osinfo driver to display your name (student name) when the minor device number is 1. (Note that the system must display the instructor information when the minor device number is 0.)**

**Create a new device file (/dev/osinfo1). Show the expected result to your instructor. The content of /dev/osinfo1 must display your group information.**

เนื่องจากว่าใน checkpoint นี้ จะต้องแสดงชื่อของสมาชิกภายในกลุ่ม จึงต้องมีการแก้ **osinfo.c** ดังนี้

จากเดิมที่ส่วนนี้ของ code เป็นแบบนี้

**static const char f\_data0[] =**

**"0:CP ENG CU OS 2022S2 - Instructors\n"**

**"1:\tVeera Muangsin, Ph.D.\n"**

**"2:\tKrerk Piromsopa, Ph.D.\n"**

**"3:\tThongchai Rojkangsadan\n";**

จะต้องเติมชื่อสมาชิกในกลุ่มเพิ่มเป็นดังนี้

**static const char f\_data0[] =**

**"0:CP ENG CU OS 2022S2 - Instructors\n"**

**"1:\tVeera Muangsin, Ph.D.\n"**

**"2:\tKrerk Piromsopa, Ph.D.\n"**

**"3:\tThongchai Rojkangsadan\n";**

**/\* ----- ADD YOUR GROUP MEMBERS HERE ----- \*/**

**static const char f\_data1[] =**

**"0:CP ENG CU OS 2022S2 - Students, Group Name: LigmaBoy\n"**

**"1:\t6630333721 Sippapas Chavanont\n"**

**"2:\t6631331621 Naytipat Phothipan\n"**

**"3:\t6632200221 Worralop Srichainont\n";**

**/\* --------------------------------------- \*/**

และจากเดิมที่ส่วนนี้ของ code เป็นแบบนี้

**static int device\_open(struct inode \*inode, struct file \*file)**

**{**

**if (dev\_open)**

**return -EBUSY;**

**dev\_open++;**

**printk(KERN\_INFO "dev minor %d\n", MINOR(inode->i\_rdev));**

**f\_ptr = (char \*)f\_data0;**

**// lock module**

**try\_module\_get(THIS\_MODULE);**

**return 0;**

**}**

จะต้องแก้ในส่วนนี้ให้เป็นแบบนี้ จึงจะสามารถแสดงชื่อของสมาชิกภายในกลุ่มได้

**static int device\_open(struct inode \*inode, struct file \*file)**

**{**

**if (dev\_open)**

**return -EBUSY;**

**dev\_open++;**

**/\* ---------- MODIFY THIS ---------- \*/**

**int minor = MINOR(inode->i\_rdev);**

**printk(KERN\_INFO "dev minor %d\n", minor);**

**if(minor == 0) {**

**f\_ptr = (char \*)f\_data0;**

**}**

**else if(minor == 1) {**

**f\_ptr = (char \*)f\_data1;**

**}**

**else {**

**return -ENODEV;**

**}**

**/\* --------------------------------- \*/**

**// lock module**

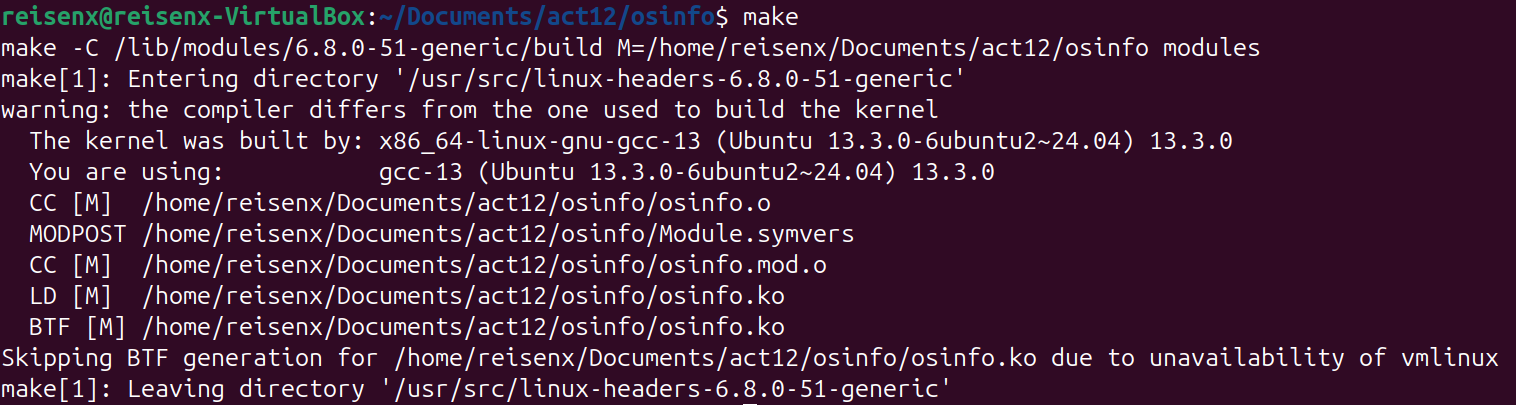
**try\_module\_get(THIS\_MODULE);**

**return 0;**

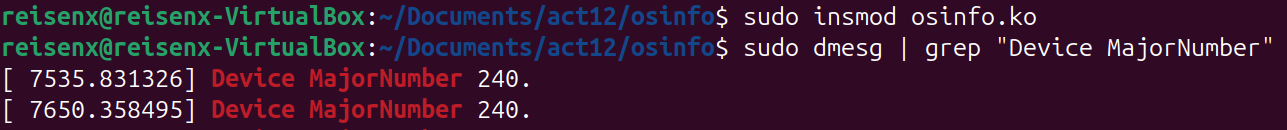
**}**

ส่วน **Makefile** ยังคงเหมือนเดิม ไม่ต้องแก้อะไร

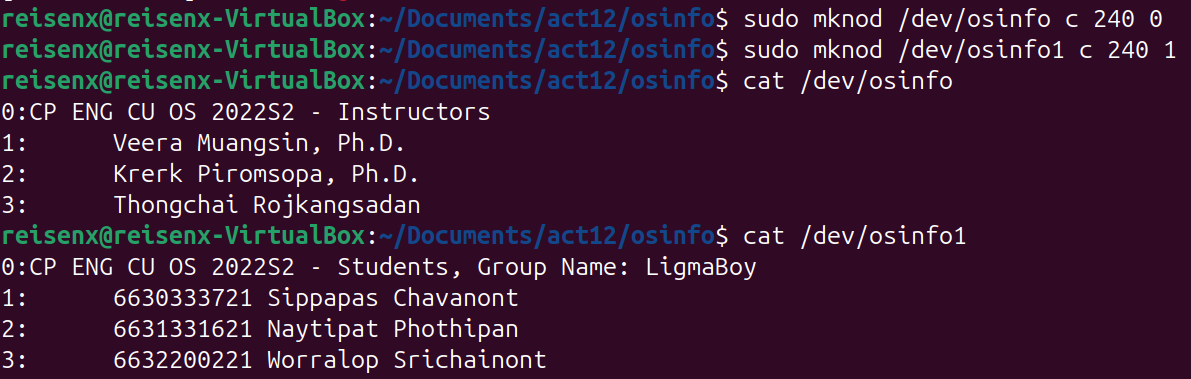
หลังจากที่แก้ไฟล์เสร็จสิ้น เราจะต้องทำการ build ใหม่ โดยใช้คำสั่ง **make**



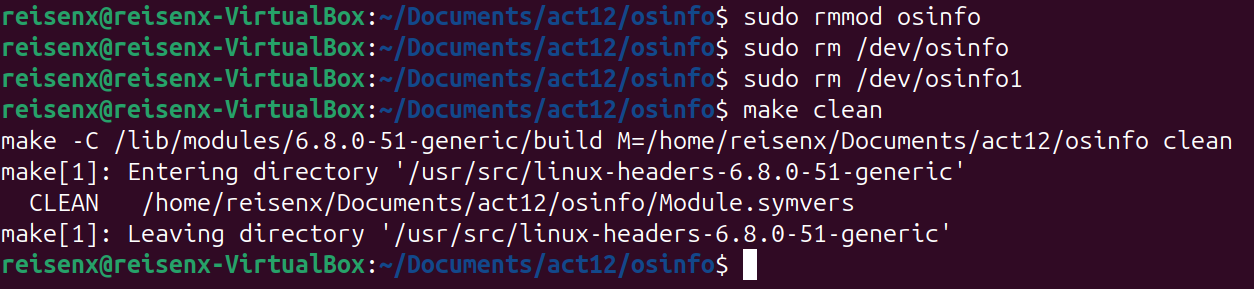
* ใช้งานคำสั่ง **sudo insmod osinfo.ko** เพื่อ load kernel module ของ osinfo
* จากนั้นตรวจสอบ MajorNumber โดยการใช้งานคำสั่ง **sudo dmesg | grep “Device MajorNumber”**



* ใช้งานคำสั่ง **sudo mknod /dev/osinfo c 240 0** เพื่อสร้าง Character Device โดยมี minor device number เท่ากับ 0 ชื่อว่า osinfo ในโฟลเดอร์ dev
* ใช้งานคำสั่ง **sudo mknod /dev/osinfo1 c 240 1** เพื่อสร้าง Character Device โดยมี minor device number เท่ากับ 1 ชื่อว่า osinfo1 ในโฟลเดอร์ dev
* แสดงผลไฟล์ Character Device ที่ถูกสร้างขึ้นมาด้วยคำสั่ง **cat /dev/osinfo** ซึ่งแสดงผลเป็นรายชื่ออาจารย์เนื่องจากมี minor device number เท่ากับ 0
* แสดงผลไฟล์ Character Device ที่ถูกสร้างขึ้นมาด้วยคำสั่ง **cat /dev/osinfo1** ซึ่งแสดงผลเป็นรายชื่อนิสิตภายในกลุ่มเนื่องจากมี minor device number เท่ากับ 1



* ใช้งานคำสั่ง **sudo rmmod osinfo** เพื่อนำ kernel module ของ osinfo ออกไปจากระบบปฏิบัติการ (unload)
* ลบโฟล์เดอร์และไฟล์ที่ถูกสร้างขึ้นมา (/dev/osinfo และ /dev/osinfo1) ด้วยคำสั่ง **sudo rm /dev/osinfo** และ **sudo rm /dev/osinfo1** ตามลำดับ
* สุดท้ายให้ใช้งานคำสั่ง **make clean** เพื่อนำไฟล์ที่ไม่จำเป็นออกไปจากเครื่อง (clean)



**Checkpoint #4**

**In this exercise, you are asked to design a character device driver that will show the vendor ID, features, and serial number of your processor.**

**The function for taking the vendor ID, features and serial number is provided.**

**Create the /dev/cpuinfo device file. The content must display the CPUID of your processor.**

**(Note that printf cannot be used within the kernel. You have to convert the code into a character device driver.)**

จาก Note ของโจทย์จะต้องทำการเปลี่ยน code ที่โจทย์ให้มาเป็น character device driver ก่อน ซึ่งหลังจากเปลี่ยนแล้วนั้น จะได้ **cpuinfo.c** ที่มีลักษณะดังนี้

**#include <linux/module.h>**

**#include <linux/kernel.h>**

**#include <linux/fs.h>**

**#include <asm/uaccess.h>**

**#include <linux/uaccess.h>**

**/\* Needed by all modules \*/**

**/\* Needed for KERN\_INFO \*/**

**MODULE\_LICENSE("GPL");**

**MODULE\_AUTHOR("KRERK PIROMSOPA, PH.D. <Krerk.P@chula.ac.th>");**

**MODULE\_DESCRIPTION("\"cpuinfo\" Character Device");**

**#define DEVICENAME "cpuinfo"**

**static int dev\_major;**

**static int dev\_open = 0;**

**static char cpuinfo\_buffer[256];  // buffer to store output**

**static char \*f\_ptr;**

**static int device\_open(struct inode \*inode, struct file \*file)**

**{**

**if (dev\_open) return -EBUSY;**

**dev\_open++;**

**unsigned eax, ebx, ecx, edx;**

**// Vendor ID**

**eax = 0;**

**ecx = 0;**

**native\_cpuid(&eax, &ebx, &ecx, &edx);**

**char vendor[13];**

**((unsigned int \*)vendor)[0] = ebx;**

**((unsigned int \*)vendor)[1] = edx;**

**((unsigned int \*)vendor)[2] = ecx;**

**vendor[12] = '\0';**

**// Feature info**

**eax = 1;**

**native\_cpuid(&eax, &ebx, &ecx, &edx);**

**int stepping = eax & 0xF;**

**int model = (eax >> 4) & 0xF;**

**int family = (eax >> 8) & 0xF;**

**int proc\_type = (eax >> 12) & 0x3;**

**int ext\_model = (eax >> 16) & 0xF;**

**int ext\_family = (eax >> 20) & 0xFF;**

**// Serial number**

**eax = 3;**

**native\_cpuid(&eax, &ebx, &ecx, &edx);**

**// Prepare the output**

**snprintf(cpuinfo\_buffer, sizeof(cpuinfo\_buffer),**

**"Vendor ID: %s\n"**

**"Stepping: %d\n"**

**"Model: %d\n"**

**"Family: %d\n"**

**"Processor Type: %d\n"**

**"Extended Model: %d\n"**

**"Extended Family: %d\n"**

**"Serial Number: 0x%08x%08x\n",**

**vendor, stepping, model, family, proc\_type, ext\_model, ext\_family, edx, ecx);**

**f\_ptr = cpuinfo\_buffer;**

**try\_module\_get(THIS\_MODULE);**

**return 0;**

**}**

**static int device\_release(struct inode \*inode, struct file \*file)**

**{**

**dev\_open--;**

**module\_put(THIS\_MODULE);**

**return 0;**

**}**

**static ssize\_t device\_read(struct file \*filp, char \*buffer, size\_t length, loff\_t \*offset)**

**{**

**int bytes\_read = 0;**

**if (\*f\_ptr == 0)**

**return 0;**

**while (length && \*f\_ptr) {**

**put\_user(\*(f\_ptr++), buffer++);**

**length--;**

**bytes\_read++;**

**}**

**return bytes\_read;**

**}**

**static struct file\_operations dev\_fops = {**

**.read    = device\_read,**

**.open    = device\_open,**

**.release = device\_release**

**};**

**int init\_module(void)**

**{**

**printk(KERN\_INFO "cpuinfo module loaded.\n");**

**dev\_major = register\_chrdev(0, DEVICENAME, &dev\_fops);**

**if (dev\_major < 0) {**

**printk(KERN\_ALERT "Registering char device failed with %d\n", dev\_major);**

**return dev\_major;**

**}**

**printk(KERN\_INFO "Device MajorNumber %d.\n", dev\_major);**

**printk(KERN\_INFO "Use: mknod /dev/cpuinfo c %d 0\n", dev\_major);**

**return 0;**

**}**

**void cleanup\_module(void)**

**{**

**unregister\_chrdev(dev\_major, DEVICENAME);**

**printk(KERN\_INFO "cpuinfo module unloaded.\n");**

**}**

และสร้าง **Makefile** โดยมีลักษณะดังนี้

**obj-m += cpuinfo.o**

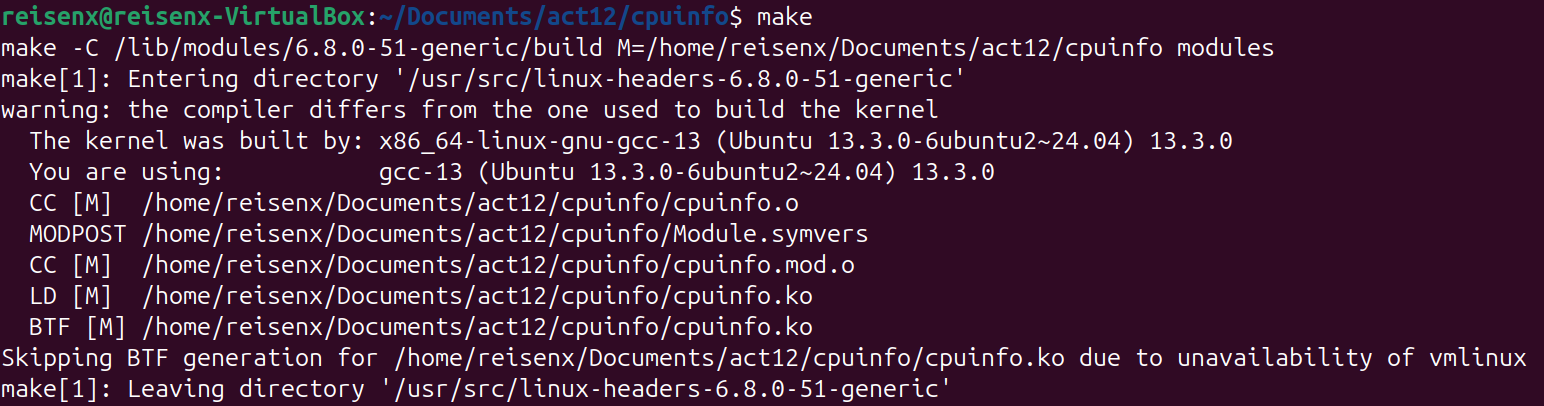
**all:**

**make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules**

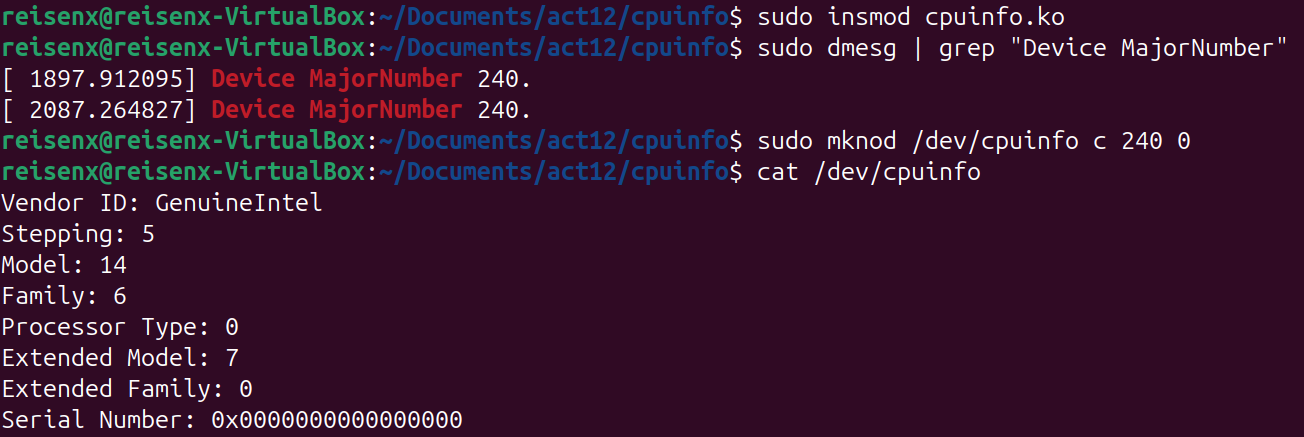
**clean:**

**make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean**

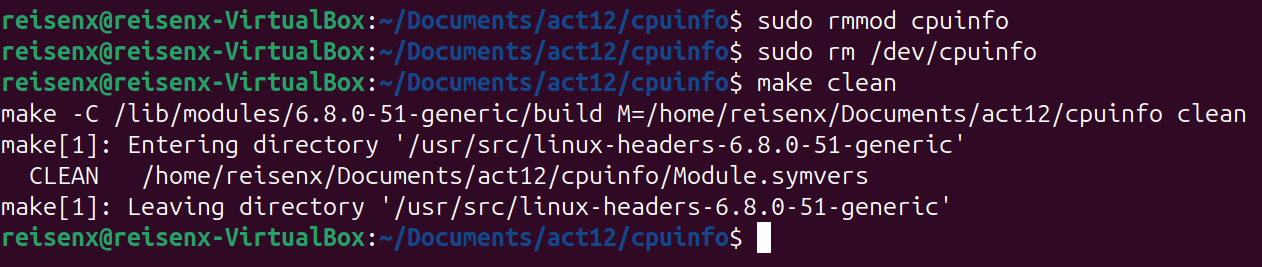
ใช้งานคำสั่ง **make** เพื่อทำให้ Makefile ทำงาน และ build ไฟล์ที่จำเป็นต่อการใช้งาน



* ใช้งานคำสั่ง **sudo insmod cpuinfo.ko** เพื่อ load kernel module ของ cpuinfo
* จากนั้นตรวจสอบ MajorNumber โดยการใช้งานคำสั่ง **sudo dmesg | grep “Device MajorNumber”**
* ใช้งานคำสั่ง **sudo mknod /dev/cpuinfo c 240 0** เพื่อสร้าง Character Device ชื่อว่า cpuinfo ในโฟลเดอร์ dev
* แสดงผลไฟล์ Character Device ที่ถูกสร้างขึ้นมาด้วยคำสั่ง **cat /dev/cpuinfo**



* ใช้งานคำสั่ง **sudo rmmod cpuinfo** เพื่อนำ kernel module ของ cpuinfo ออกไปจากระบบปฏิบัติการ (unload)
* ลบโฟล์เดอร์และไฟล์ที่ถูกสร้างขึ้นมา (/dev/cpuinfo) ด้วยคำสั่ง **sudo rm /dev/cpuinfo**
* สุดท้ายให้ใช้งานคำสั่ง **make clean** เพื่อนำไฟล์ที่ไม่จำเป็นออกไปจากเครื่อง (clean)



**Checkpoint #5**

**Create a character device (cpsysinfo) as a kernel module. This character device must provide information about your operating systems as follows**

**Minor 0: active processes on your operating system.**

**Minor 1: Amount of memory**

**Create the /dev/cp-psinfo device file (Minor 0). The file must provide pid, command of all processes.**

ในขั้นตอนนี้ จะต้องสร้างไฟล์ **cpsysinfo.c** ที่มีลักษณะดังนี้

**#include <linux/init.h>**

**#include <linux/module.h>**

**#include <linux/kernel.h>**

**#include <linux/fs.h>**

**#include <linux/uaccess.h>**

**#include <linux/device.h>**

**#include <linux/sched/signal.h>  // for\_each\_process**

**#include <linux/mm.h>            // for si\_meminfo**

**#include <linux/slab.h>          // for kzalloc**

**#define DEVICE\_NAME "cpsysinfo"**

**#define CLASS\_NAME "cpclass"**

**MODULE\_LICENSE("GPL");**

**MODULE\_AUTHOR("KRERK PIROMSOPA, PH.D. <Krerk.P@chula.ac.th>");**

**MODULE\_DESCRIPTION("\"cpsysinfo\" Character Device");**

**static int majorNumber;**

**static struct class\* cpClass = NULL;**

**static struct device\* cpDevice0 = NULL;**

**static struct device\* cpDevice1 = NULL;**

**static char \*info\_buffer;**

**static int info\_size;**

**static int already\_read = 0;**

**#define TO\_KB(pages) ((pages) \* (PAGE\_SIZE / 1024))**

**static ssize\_t dev\_read(struct file \*filep, char \*buffer, size\_t len, loff\_t \*offset) {**

**int bytes\_to\_copy;**

**if (already\_read) {**

**already\_read = 0;**

**return 0;**

**}**

**bytes\_to\_copy = min(len, (size\_t)info\_size);**

**if (copy\_to\_user(buffer, info\_buffer, bytes\_to\_copy)) {**

**return -EFAULT;**

**}**

**already\_read = 1;**

**return bytes\_to\_copy;**

**}**

**static int dev\_open(struct inode \*inodep, struct file \*filep) {**

**int minor = MINOR(inodep->i\_rdev);**

**struct task\_struct \*task;**

**if (info\_buffer)**

**kfree(info\_buffer);**

**info\_buffer = kzalloc(8192, GFP\_KERNEL);**

**if (!info\_buffer)**

**return -ENOMEM;**

**info\_size = 0;**

**already\_read = 0;**

**if (minor == 0) {**

**// Process list**

**for\_each\_process(task) {**

**info\_size += snprintf(info\_buffer + info\_size, 8192 - info\_size,**

**"%d,%s\n", task->pid, task->comm);**

**if (info\_size >= 8192 - 100)**

**break;**

**}**

**} else if (minor == 1) {**

**// Memory info**

**struct sysinfo i;**

**si\_meminfo(&i);**

**unsigned long mem\_total = TO\_KB(i.totalram);**

**unsigned long mem\_free = TO\_KB(i.freeram);**

**unsigned long mem\_available = TO\_KB(i.freeram + i.bufferram);**

**info\_size = snprintf(info\_buffer, 8192,**

**"MemTotal: %lu kB\n"**

**"MemFree: %lu kB\n"**

**"MemAvailable: %lu kB\n",**

**mem\_total, mem\_free, mem\_available);**

**} else {**

**snprintf(info\_buffer, 8192, "Invalid minor number: %d\n", minor);**

**info\_size = strlen(info\_buffer);**

**}**

**return 0;**

**}**

**static int dev\_release(struct inode \*inodep, struct file \*filep) {**

**return 0;**

**}**

**static struct file\_operations fops = {**

**.open = dev\_open,**

**.read = dev\_read,**

**.release = dev\_release,**

**};**

**static int \_\_init cpsysinfo\_init(void) {**

**majorNumber = register\_chrdev(0, DEVICE\_NAME, &fops);**

**if (majorNumber < 0) {**

**printk(KERN\_ALERT "Failed to register char device\n");**

**return majorNumber;**

**}**

**cpClass = class\_create(CLASS\_NAME);  // For Linux 6.8+**

**if (IS\_ERR(cpClass)) {**

**unregister\_chrdev(majorNumber, DEVICE\_NAME);**

**return PTR\_ERR(cpClass);**

**}**

**cpDevice0 = device\_create(cpClass, NULL, MKDEV(majorNumber, 0), NULL, DEVICE\_NAME);**

**cpDevice1 = device\_create(cpClass, NULL, MKDEV(majorNumber, 1), NULL, DEVICE\_NAME "1");**

**if (IS\_ERR(cpDevice0) || IS\_ERR(cpDevice1)) {**

**device\_destroy(cpClass, MKDEV(majorNumber, 0));**

**device\_destroy(cpClass, MKDEV(majorNumber, 1));**

**class\_destroy(cpClass);**

**unregister\_chrdev(majorNumber, DEVICE\_NAME);**

**return -1;**

**}**

**printk(KERN\_INFO "cpsysinfo loaded. Major number: %d\n", majorNumber);**

**printk(KERN\_INFO "Use:\n");**

**printk(KERN\_INFO "  sudo mknod /dev/cp-psinfo c %d 0\n", majorNumber);**

**printk(KERN\_INFO "  sudo mknod /dev/cp-meminfo c %d 1\n", majorNumber);**

**return 0;**

**}**

**static void \_\_exit cpsysinfo\_exit(void) {**

**if (info\_buffer)**

**kfree(info\_buffer);**

**device\_destroy(cpClass, MKDEV(majorNumber, 0));**

**device\_destroy(cpClass, MKDEV(majorNumber, 1));**

**class\_unregister(cpClass);**

**class\_destroy(cpClass);**

**unregister\_chrdev(majorNumber, DEVICE\_NAME);**

**printk(KERN\_INFO "cpsysinfo unloaded\n");**

**}**

**module\_init(cpsysinfo\_init);**

**module\_exit(cpsysinfo\_exit);**

และสร้าง **Makefile** โดยมีลักษณะดังนี้

**obj-m += cpsysinfo.o**

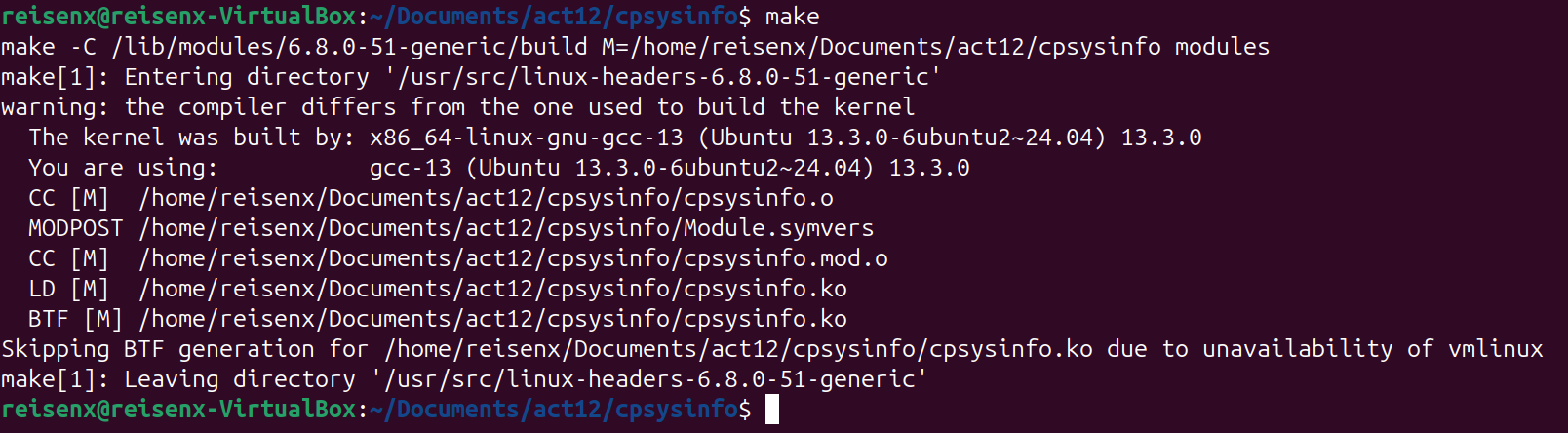
**all:**

**make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules**

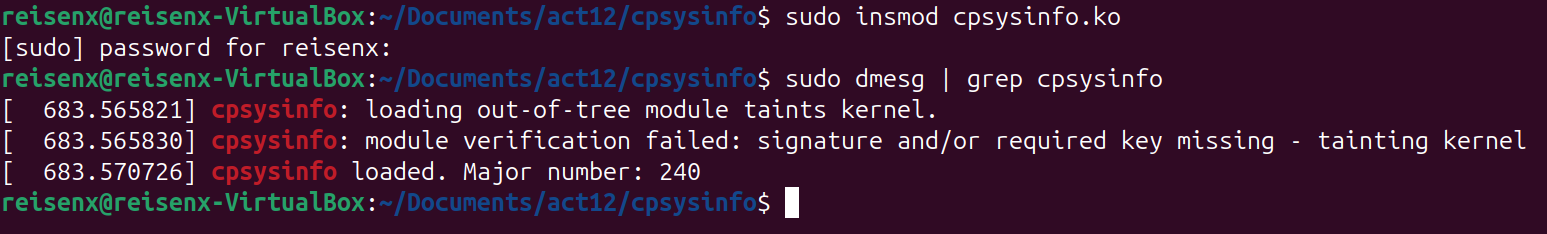
**clean:**

**make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean**

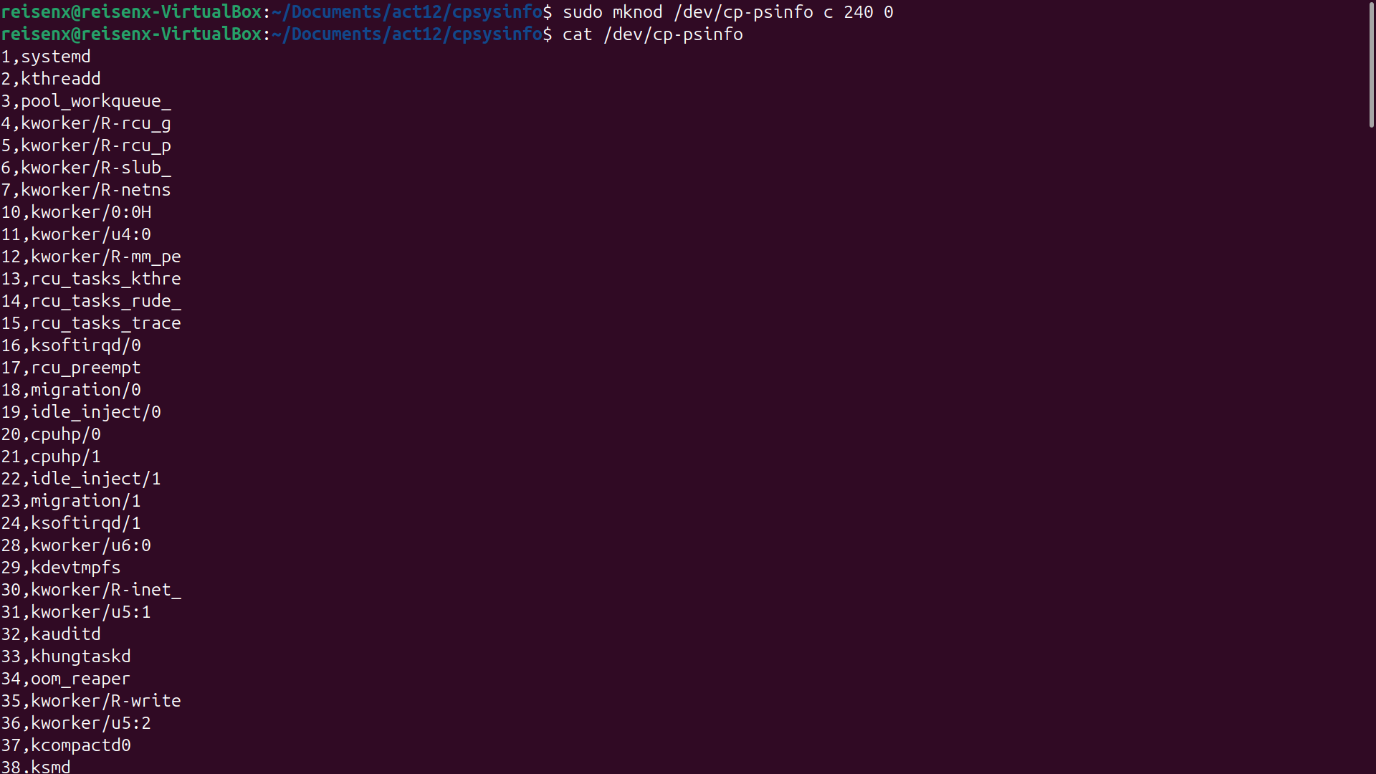
ใช้งานคำสั่ง **make** เพื่อทำให้ Makefile ทำงาน และ build ไฟล์ที่จำเป็นต่อการใช้งาน



* ใช้งานคำสั่ง **sudo insmod cpsysinfo.ko** เพื่อ load kernel module ของ cpsysinfo
* จากนั้นตรวจสอบ MajorNumber โดยการใช้งานคำสั่ง **sudo dmesg | grep cpsysinfo**



* ใช้งานคำสั่ง **sudo mknod /dev/cp-psinfo c 240 0** เพื่อสร้าง Character Device ชื่อว่า cp-psinfo ในโฟลเดอร์ dev
* แสดงผลไฟล์ Character Device ที่ถูกสร้างขึ้นมาด้วยคำสั่ง **cat /dev/cp-psinfo**
* เนื่องจากว่า device ที่สร้างมามี minor device number เท่ากับ 0 ดังนั้นมันจะแสดงผลรายการ process ที่ active ในระบบปฏิบัติการ ณ ขณะนั้น

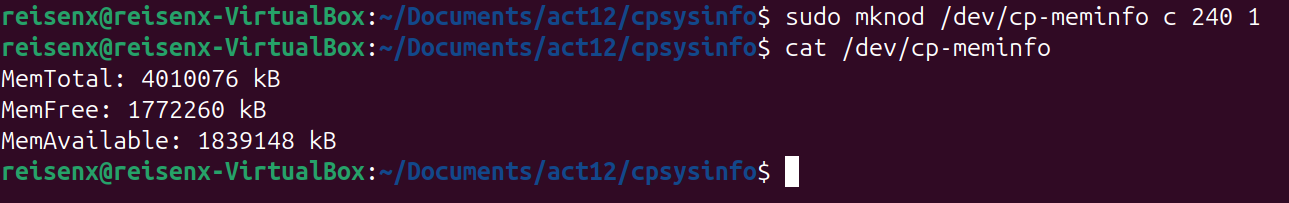


**Checkpoint #6**

**Create the /dev/cp-meminfo device file (Minor 1). The file must provide the total amount of available memory. (The result should look similar to /proc/meminfo.)**

ใน checkpoint นี้สามารถทำต่อจาก checkpoint ที่แล้วได้เลย เพียงใช้คำสั่งข้างล่างนี้ก็เป็นอันเสร็จสิ้น

* ใช้งานคำสั่ง **sudo mknod /dev/cp-meminfo c 240 1** เพื่อสร้าง Character Device ชื่อว่า cp-meminfo ในโฟลเดอร์ dev
* แสดงผลไฟล์ Character Device ที่ถูกสร้างขึ้นมาด้วยคำสั่ง **cat /dev/cp-meminfo**
* เนื่องจากว่า device ที่สร้างมามี minor device number เท่ากับ 1 ดังนั้นมันจะแสดงผลขนาดหน่วยความจำของระบบปฏิบัติการ



* ใช้งานคำสั่ง **sudo rmmod cpsysinfo** เพื่อนำ kernel module ของ cpsysinfo ออกไปจากระบบปฏิบัติการ (unload)
* ลบโฟล์เดอร์และไฟล์ที่ถูกสร้างขึ้นมา (/dev/cp-psinfo และ /dev/cp-meminfo) ด้วยคำสั่ง **sudo rm /dev/cp-psinfo** และ **sudo rm /dev/cp-meminfo** ตามลำดับ
* สุดท้ายให้ใช้งานคำสั่ง **make clean** เพื่อนำไฟล์ที่ไม่จำเป็นออกไปจากเครื่อง (clean)

