

# System Programming Lab #4 Kernel Lab

---

2023-05-02

sp-tas

# Lab Assignment #4 : Kernel Lab

- Download skeleton code & pdf from eTL
  - kernellab-handout.tar, kernellab-handout.pdf
- Hand In – Your Implementation
  - Upload your files eTL
    - 압축파일 양식 : [학번]\_kernellab.zip
    - Ex) 2023-12345\_kernellab.zip
  - A zip file should include
    - (1) a zip file of your implementation directory (2) report
      - 압축파일 양식 : [학번]\_kernellab.zip      eg) 2023-12345\_kernellab.zip
      - Report 양식 : [학번]\_kernellab\_report.pdf
- Please, **READ** the Hand-out and Lab material thoroughly!
- Assigned : 5.2(tue)
- Deadline : 5.22(mon), 23:59
- Delay : same as before
- Only use your own VM(**Do not use server machine**)

# Lab Assignment #4 : Kernel Lab

---

- Implementation
  - (part #1) Tracing process tree from process id
  - (part #2) Finding physical address using virtual address

# Linux Kernel Programming

---

- Kernel : “Core of OS”
- Why?
  - Scheduling
  - Cache replacement
  - File system
  - Get kernel information
  - Or other reasons

# Linux Kernel Programming

- How to modify kernel?

>> modify source and compile your own kernel

- Good idea, but it is difficult and takes too much time
  - It may take 1-2 hours to compile kernel in your VM.

- Any simpler way?

>> **Make kernel modules(Kernel Lab)**

# Linux Kernel Module

- What is kernel module?
  - A module is pieces of code that can be **loaded and unloaded into the kernel** upon demand.
  - Can **use privileged instructions without system calls**, because a kernel module is loaded and executed within a kernel.
- Need special compile
  - Kernel module is not compiled with general gcc
  - It needs kernel specific compile tools
- Module load / unload command
  - **Load** `$ sudo insmod < module_name.ko >`
  - **Unload** `$ sudo rmmod < module_name >`
  - **Module list** `$ sudo lsmod`

# Kernel Module Programming

- Kernel module convention

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

MODULE_LICENSE("GPL");

static int __init init_my_module(void)
{
    // Running when this module is inserted to Kernel
}

static void __exit exit_my_module(void)
{
    // Running when this module is removed from Kernel
}

module_init(init_my_module);
module_exit(exit_my_module);
```

# Kernel Module Programming

- `module_init(func)`
  - 드라이버의 초기화 진입점
  - `func` : 커널 부트나 모듈 삽입 시 실행할 함수
  - `__init` : 모듈 삽입 시 한 번만 실행하게 하는 매크로
- `module_exit(func)`
  - 드라이버의 종료 진입점
  - `func` : 드라이버가 제거될 때 실행할 함수
  - `__exit` : 종료 시에 실행하게 하는 매크로

\* `__init`, `__exit`은 특정 섹션에 넣어 놓고 필요 없어지면 메모리에서 해제하는 식으로 동작

\* 설명 : <kernel source>/include/linux/init.h



# Kernel Module Compile Example

- Makefile for kernel module

```
KDIR = /lib/modules/$(shell uname -r)/build  
  
obj-m := hello.o  
  
all :  
    $(MAKE) -C $(KDIR) M=$(PWD) modules;  
  
clean :  
    $(MAKE) -C $(KDIR) M=$(PWD) clean;
```

- What are those options?

# Kernel Module Compile Options

- \$(MAKE) : 'make' command
- -C \$(KDIR) : move to KDIR directory before running make
- M : 'M' is variable. It means where you build module or where your project is
- \$(PWD) : current directory
- obj-m: build the object file as module

```
KDIR = /lib/modules/$(shell uname -r)/build  
  
obj-m := hello.o  
  
all :  
    $(MAKE) -C $(KDIR) M=$(PWD) modules;  
  
clean :  
    $(MAKE) -C $(KDIR) M=$(PWD) clean;
```

- == KDIR로 가서, 'make modules' 명령을 실행하여 현재 디렉토리에 hello.o를 빌드하여 모듈로 만들어라.

# Make interface!

- App needs interface to kernel. Why?
  - To get kernel info
  - To send app info
  - To do HW something(read/write..)
  - Or any other things
- Then how?
  - System call(need to modify kernel itself)
  - **Debug Files System**

# Debugfs

- **Debug File System (debugfs)**
  - Special file system(RAM –based)
  - Designed for debugging purposes
  - Simple way for kernel developers to make information available to user space
  - Has no rules at all.  
The developers can put any information they want.
  - Supports simple user-to-kernel interfaces in Linux Kernel Modules.

# Debugfs API

- APIs in *<linux/debugfs.h>*
- **struct dentry\*** debugfs\_create\_dir(**const char** \*name, **struct dentry** \*parent)
  - make <name> directory under <parent> directory
  - If NULL in parent, make directory under root directory
- **struct dentry\*** debugfs\_create\_file(**const char** \*name, **umode\_t** mode, **struct dentry** \*parent, **void** \*data, **const struct file\_operations** \*fops)
  - make <name> file under <parent> directory with <mode> permission(e.g. 0644).
  - when you do file operation(open/r/w/close ...) use <fops>
  - caller gets <data>. inode.i\_privte point this on open() call
- **struct dentry\*** debugfs\_remove\_recursive(**struct dentry** \*dentry)
  - recursively remove <dentry> file/dir.
- dentry? file\_operations?
- \*Ref: [DebugFS — The Linux Kernel documentation](#)

# dentry

---

- “Directory Entry”
- Connect file name with inode
- Simply, a file or a directory in this lab
- Defined in *<kernel source>/include/linux/dcache.h*

# file\_operations

- *struct file*
  - **Abstraction** on all types of files (i.e. regular file, dev file)
  - file inode, name, type, size etc.
  - **file\_operations**
- mapping file operation(.read) to your function(file\_read)

```
struct file_operations Fops = {  
    .read = file_read,  
    .write = file_write,  
    .open = file_open,  
    .release = file_close,  
};
```
- Defined in *<kernel source>/include/linux/fs.h*

# file\_operations

- C language does not support struct inheritance
- For using **same interface** (syscall) for different files
- ex) syscall *write()* (*file->fops->write()* is called)
  - ext4\_write() for file in ext4
  - f2fs\_write() for file in f2fs
  - blkdev\_write() for file block device

\*(this is just an example for simple explanation)



# Prepare your own development environment

- Part #1: ptree module compile preparation
- Use 'dmesg' command

sptest [실경 중] - Oracle VM VirtualBox

파일 마신 보기 입력 장치 도움말

root@spta-VirtualBox: /home/spta

```
root@spta-VirtualBox: /home/spta/Downloads/CSAP_KernelLab/skeleton/ptree# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 16.04.1 LTS
Release: 16.04
Codename: xenial

root@spta-VirtualBox: /home/spta/Downloads/CSAP_KernelLab/skeleton/ptree# uname -ar
Linux spta-VirtualBox: 4.4.0-31-generic #50-Ubuntu SMP Wed Jul 13 00:07:12 UTC 2016 x86_64 x86_64 GNU/Linux

root@spta-VirtualBox: /home/spta/Downloads/CSAP_KernelLab/skeleton/ptree# make
make -C /lib/modules/4.4.0-31-generic/build M=/home/spta/Downloads/CSAP_KernelLab/skeleton/ptree module
make[1]: Entering directory '/usr/src/linux-headers-4.4.0-31-generic'
  CC [M] /home/spta/Downloads/CSAP_KernelLab/skeleton/ptree/dbfs_ptree.o
/home/spta/Downloads/CSAP_KernelLab/skeleton/ptree/dbfs_ptree.c: In function 'write_pid_to_input':
/home/spta/Downloads/CSAP_KernelLab/skeleton/ptree/dbfs_ptree.c:16:15: warning: unused variable 'input
_pid' [-Wunused-variable]
     pid_t input_pid;
     ^
/home/spta/Downloads/CSAP_KernelLab/skeleton/ptree/dbfs_ptree.c: At top level:
/home/spta/Downloads/CSAP_KernelLab/skeleton/ptree/dbfs_ptree.c:8:23: warning: 'dir' defined but not u
sed [-Wunused-variable]
     static struct dentry *dir, *inputdir, *ptreedir;
                        ^
/home/spta/Downloads/CSAP_KernelLab/skeleton/ptree/dbfs_ptree.c:8:29: warning: 'inputdir' defined but
not used [-Wunused-variable]
     static struct dentry *dir, *inputdir, *ptreedir;
                               ^
/home/spta/Downloads/CSAP_KernelLab/skeleton/ptree/dbfs_ptree.c:8:40: warning: 'ptreedir' defined but
not used [-Wunused-variable]
     static struct dentry *dir, *inputdir, *ptreedir;
                                   ^
/home/spta/Downloads/CSAP_KernelLab/skeleton/ptree/dbfs_ptree.c:9:28: warning: 'curr' defined but not
used [-Wunused-variable]
     static struct task_struct *curr;
                        ^
Building modules, stage 2.
MODPOST 1 modules
  CC      /home/spta/Downloads/CSAP_KernelLab/skeleton/ptree/dbfs_ptree.ko
  LD [M]   /home/spta/Downloads/CSAP_KernelLab/skeleton/ptree/dbfs_ptree.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.4.0-31-generic'
sudo insmod dbfs_ptree.ko
root@spta-VirtualBox: /home/spta/Downloads/CSAP_KernelLab/skeleton/ptree# make clean
make -C /lib/modules/4.4.0-31-generic/build M=/home/spta/Downloads/CSAP_KernelLab/skeleton/ptree clean
;
make[1]: Entering directory '/usr/src/linux-headers-4.4.0-31-generic'
  CLEAN   /home/spta/Downloads/CSAP_KernelLab/skeleton/ptree/.tmp_versions
  CLEAN   /home/spta/Downloads/CSAP_KernelLab/skeleton/ptree/Module.symvers
make[1]: Leaving directory '/usr/src/linux-headers-4.4.0-31-generic'
sudo rmmod dbfs_ptree.ko
root@spta-VirtualBox: /home/spta/Downloads/CSAP_KernelLab/skeleton/ptree#
```

```
[ 5.265557] systemd-journald[222]: Received request to flush runtime journal from PID 1
[ 5.441991] audit: type=1400 audit(1555842325.756:2): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/lib/lightdm/lightdm-guest-session" pid=413 comm="apparmor_parser"
[ 5.441995] audit: type=1400 audit(1555842325.756:3): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/lib/lightdm/lightdm-guest-session/chromium" pid=413 comm="apparmor_pa
rser"
[ 5.445794] audit: type=1400 audit(1555842325.760:4): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/sbin/dhclient" pid=415 comm="apparmor_parser"
[ 5.445797] audit: type=1400 audit(1555842325.760:5): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/lib/NetworkManager/nm-dhcp-client.action" pid=415 comm="apparmor_parse
r"
[ 5.445799] audit: type=1400 audit(1555842325.760:6): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/lib/NetworkManager/nm-dhcp-helper" pid=415 comm="apparmor_parser"
[ 5.445801] audit: type=1400 audit(1555842325.760:7): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/lib/connman/scripts/dhclient-script" pid=415 comm="apparmor_parser"
[ 5.463102] audit: type=1400 audit(1555842325.776:8): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/bin/evince" pid=417 comm="apparmor_parser"
[ 5.463106] audit: type=1400 audit(1555842325.776:9): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/bin/evince/sanitized helper" pid=417 comm="apparmor_parser"
[ 5.463108] audit: type=1400 audit(1555842325.776:10): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/bin/evince-previewer" pid=417 comm="apparmor_parser"
[ 5.743474] vgdvrHeartbeatInit: Setting up heartbeat to trigger every 2000 milliseconds
[ 5.743547] input: Unspecified device as /devices/pci0000:00/0000:00:04:0/input/input7
[ 5.769195] vboxguest: misc device minor 55, IRQ 20, I/O port d020, MMIO at 00000000f4000000 (size
0x400000)
[ 5.769197] vboxguest: Successfully loaded version 5.0.18_Ubuntu (interface 0x00010004)
[ 5.788740] random: nonblocking pool is initialized
[ 5.891797] [drm] Initialized drm 1.1.0 20060810
[ 5.901461] pitx4 smbus 0000:00:07:0: SMBus Host Controller at 0x4100, revision 0
[ 5.954858] AVX2 version of gcm enc/dec engaged.
[ 5.954860] AES CTR mode by8 optimization enabled
[ 5.957749] [drm] VRAM 01000000
[ 5.971386] [ITM] Zone kernel: Available graphics memory: 508136 KiB
[ 5.971389] [ITM] Initializing pool allocator
[ 5.971392] [ITM] Initializing DMA pool allocator
[ 5.976199] fbcon: vboxdrmfb (fb0) is primary device
[ 6.043385] Console: switching to colour frame buffer device 100x37
[ 6.048086] vboxvideo 0000:00:02:0: fb0: vboxdrmfb frame buffer device
[ 6.048625] [drm] Initialized vboxvideo 1.0.0 20130823 for 0000:00:02:0 on minor 0
[ 6.049011] snd_intel8x0 0000:00:05:0: disable (unknown or VT-d) VM optimization
[ 6.061349] intel_rapl: no valid rapl domains found in package 0
[ 6.429932] snd_intel8x0 0000:00:05:0: white list rate for 1028:0177 is 48000
[ 6.775637] Adding 1046524k swap on /dev/sda5. Priority:-1 extents:1 across:1046524k FS
[ 8.085821] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[ 8.087517] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[ 8.088602] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[ 8.088867] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[ 8.758092] floppy0: no floppy controllers found
[ 8.758125] work still pending
[ 271.472653] dbfs_ptree: module verification failed: signature and/or required key missing - tainti
ng kernel
[ 271.472785] dbfs_ptree module initialize done
[ 275.029860] dbfs_ptree module exit
```

# Prepare your own development environment

- Part #2: paddr module compile preparation

sptest [실행 중] - Oracle VM VirtualBox

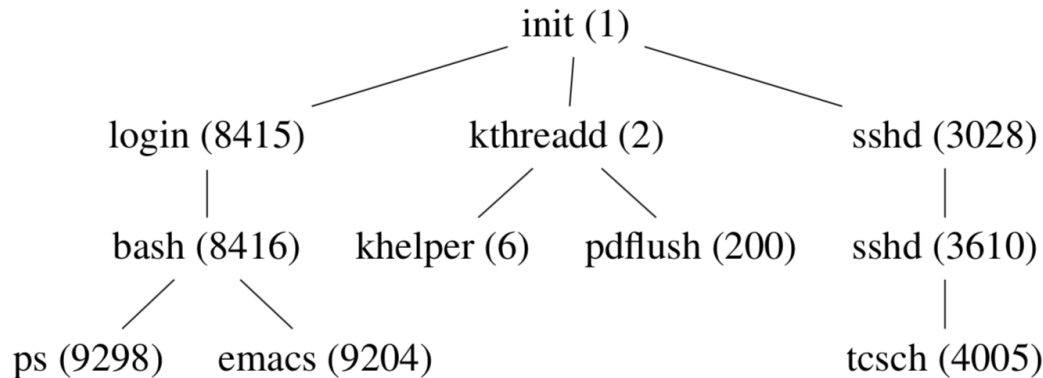
파일 편집 보기 입력 장치 도움말

```
Terminal File Edit View Search Terminal Help
root@spta-VirtualBox: /home/spta/Downloads/CSAP_KernelLab/skeleton/paddr # ls -ls -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 16.04.1 LTS
Release: 16.04
Codename: xenial
root@spta-VirtualBox: /home/spta/Downloads/CSAP_KernelLab/skeleton/paddr # uname -ar
Linux spta-VirtualBox 4.4.0-31-generic #50-Ubuntu SMP Wed Jul 13 00:07:12 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
root@spta-VirtualBox: /home/spta/Downloads/CSAP_KernelLab/skeleton/paddr # make
make -C /lib/modules/4.4.0-31-generic/build M=/home/spta/Downloads/CSAP_KernelLab/skeleton/paddr module
make[1]: Entering directory '/usr/src/linux-headers-4.4.0-31-generic'
CC [M] /home/spta/Downloads/CSAP_KernelLab/skeleton/paddr/dbfs_paddr.o
/home/spta/Downloads/CSAP_KernelLab/skeleton/paddr/dbfs_paddr.c: In function 'read_output':
/home/spta/Downloads/CSAP_KernelLab/skeleton/paddr/dbfs_paddr.c:18:1: warning: no return statement in
function returning non-void [-Wreturn-type]
}
/home/spta/Downloads/CSAP_KernelLab/skeleton/paddr/dbfs_paddr.c: At top level:
/home/spta/Downloads/CSAP_KernelLab/skeleton/paddr/dbfs_paddr.c:9:23: warning: 'dir' defined but not u
sed [-Wunused-variable]
static struct dentry *dir, *output;
/home/spta/Downloads/CSAP_KernelLab/skeleton/paddr/dbfs_paddr.c:9:29: warning: 'output' defined but no
t used [-Wunused-variable]
static struct dentry *dir, *output;
/home/spta/Downloads/CSAP_KernelLab/skeleton/paddr/dbfs_paddr.c:10:28: warning: 'task' defined but not
used [-Wunused-variable]
static struct task_struct *task;
/home/spta/Downloads/CSAP_KernelLab/skeleton/paddr/dbfs_paddr.c:12:16: warning: 'read_output' defined
but not used [-Wunused-function]
static ssize_t read_output(struct file *fp,
Building modules, stage 2.
MODPOST 1 modules
CC /home/spta/Downloads/CSAP_KernelLab/skeleton/paddr/dbfs_paddr.mod.o
LD [M] /home/spta/Downloads/CSAP_KernelLab/skeleton/paddr/dbfs_paddr.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.4.0-31-generic'
gcc -o app app.c;
sudo insmod dbfs_paddr.ko
root@spta-VirtualBox: /home/spta/Downloads/CSAP_KernelLab/skeleton/paddr # make clean
make -C /lib/modules/4.4.0-31-generic/build M=/home/spta/Downloads/CSAP_KernelLab/skeleton/paddr clean
make[1]: Entering directory '/usr/src/linux-headers-4.4.0-31-generic'
CLEAN /home/spta/Downloads/CSAP_KernelLab/skeleton/paddr/.tmp_versions
CLEAN /home/spta/Downloads/CSAP_KernelLab/skeleton/paddr/Module.symvers
make[1]: Leaving directory '/usr/src/linux-headers-4.4.0-31-generic'
rm app;
sudo rmmod dbfs_paddr.ko
root@spta-VirtualBox: /home/spta/Downloads/CSAP_KernelLab/skeleton/paddr #
```

```
rofile="unconfined" name="/usr/lib/lightdm/lightdm-guest-session" pid=413 comm="apparmor_parser"
[ 5.441995] audit: type=1400 audit(1555842325.756:3): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/lib/lightdm/lightdm-guest-session//chromium" pid=413 comm="apparmor_pa
rser"
[ 5.445794] audit: type=1400 audit(1555842325.760:4): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/sbin/dhclient" pid=415 comm="apparmor_parser"
[ 5.445797] audit: type=1400 audit(1555842325.760:5): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/lib/NetworkManager/nm-dhcp-client.action" pid=415 comm="apparmor_parse
r"
[ 5.445799] audit: type=1400 audit(1555842325.760:6): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/lib/NetworkManager/nm-dhcp-helper" pid=415 comm="apparmor_parser"
[ 5.445801] audit: type=1400 audit(1555842325.760:7): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/lib/connman/scripts/dhclient-script" pid=415 comm="apparmor_parser"
[ 5.463102] audit: type=1400 audit(1555842325.776:8): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/bin/evince" pid=417 comm="apparmor_parser"
[ 5.463106] audit: type=1400 audit(1555842325.776:9): apparmor="STATUS" operation="profile_load" p
rofile="unconfined" name="/usr/bin/evince//sanitized_helper" pid=417 comm="apparmor_parser"
[ 5.463108] audit: type=1400 audit(1555842325.776:10): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/bin/evince-previewer" pid=417 comm="apparmor_parser"
[ 5.743474] vboxdrvHeartBeatInit: Setting up heartbeat to trigger every 2000 milliseconds
[ 5.743547] Input: Unspecified device as /devices/pci0000:00/0000:00:04.0/input/input7
[ 5.769195] vboxguest: msc device minor 55, IRQ 20, I/O port d020, MMIO at 00000000f0400000 (size
0x400000)
[ 5.769197] vboxguest: Successfully loaded version 5.0.18_Ubuntu (interface 0x00010004)
[ 5.788740] random: nonblocking pool is initialized
[ 5.891797] [drm] Initialized drm 1.1.0 20060810
[ 5.901461] piix4_smbus 0000:00:07.0: SMBus Host Controller at 0x4100, revision 0
[ 5.954858] AVX2 version of gcm_enc/dec engaged.
[ 5.954860] AES CTR mode by8 optimization enabled
[ 5.957749] [drm] VRAM 01000000
[ 5.971386] [TTM] Zone kernel: Available graphics memory: 508136 kiB
[ 5.971389] [TTM] Initializing pool allocator
[ 5.971392] [TTM] Initializing DMA pool allocator
[ 5.976199] fbcon: vboxdrmfb (fb0) is primary device
[ 6.043385] Console: switching to colour frame buffer device 100x37
[ 6.048086] vboxvideo 0000:00:02.0: fb0: vboxdrmfb frame buffer device
[ 6.048625] [drm] Initialized vboxvideo 1.0.0 20130823 for 0000:00:02.0 on minor 0
[ 6.049011] snd_intel8x0 0000:00:05.0: disable (unknown or VT-d) VM optimization
[ 6.061349] intel_rapl: no valid rapl domains found in package 0
[ 6.062932] snd_intel8x0 0000:00:05.0: white list rate for 1028:0177 is 48000
[ 6.775637] Adding 1046524k swap on /dev/sda5. Priority:-1 extents:1 across:1046524k FS
[ 8.085821] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[ 8.087517] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[ 8.088602] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[ 8.088607] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[ 8.758092] floppy0: no floppy controllers found
[ 8.758125] work still pending
[ 271.472653] dbfs_ptree: module verification failed: signature and/or required key missing - tainti
ng kernel
[ 271.472785] dbfs_ptree module initialize done
[ 344.645272] dbfs_ptree module exit
[ 344.645272] dbfs_paddr module initialize done
[ 348.375903] dbfs_paddr module exit
```

# Kernel lab part #1 – tracing parent process tree

- Trace process from leaf to init process



- Spec
  - Input : [input process id]
  - Output : list of [process name] [process id]

Ex) input : 9204                  output : init(1)  
                                        login (8415)  
                                        bash (8416)  
                                        emacs (9204)

# Kernel lab part #1 – tracing parent process tree

- Testing

- Get root access `user# sudo su`
- Go to ptree dir `root# cd /sys/kernel/debug/ptree`
- Show current process `root# ps`
- Write input PID to file `root# echo [input process id] >> input`
- Read ptree file `root# cat ptree`

- Example output

```
unix> cat ptree
init (1)
xfce4-panel (2306)
xfce4-terminal (2408)
bash (2413)
sudo (2881)
```

# With Skeleton Code

```
static int __init dbfs_module_init(void)
```

<- executed when module is inserted

```
{
```

```
    // Implement init module code
```

```
    dir = debugfs_create_dir("ptree", NULL);
```

```
    if (!dir) {
```

```
        printk("Cannot create ptree dir\n");
```

```
        return -1;
```

```
    }
```

```
    inputdir = debugfs_create_file("input", , , , );
```

<- file to read input

```
    ptreedir = debugfs_create_dir("ptree", , , );
```

<- file to write output

// Find suitable debugfs API

```
    printk("dbfs_ptree module initialize done\n");
```

```
    return 0;
```

```
}
```

```
static void __exit dbfs_module_exit(void)
```

<- executed when module is deleted

```
{
```

```
    // Implement exit module code
```

```
    printk("dbfs_ptree module exit\n");
```

```
}
```

```
module_init(dbfs_module_init);
```

```
module_exit(dbfs_module_exit);
```

# With Skeleton Code

- ```
static ssize_t write_pid_to_input(struct file *fp,
                                const char __user *user_buffer,
                                size_t length,
                                loff_t *position)
{
    pid_t input_pid;

    sscanf(user_buffer, "%u", &input_pid);    <- read input pid
    //curr = // Find task_struct using input_pid. Hint: pid_task

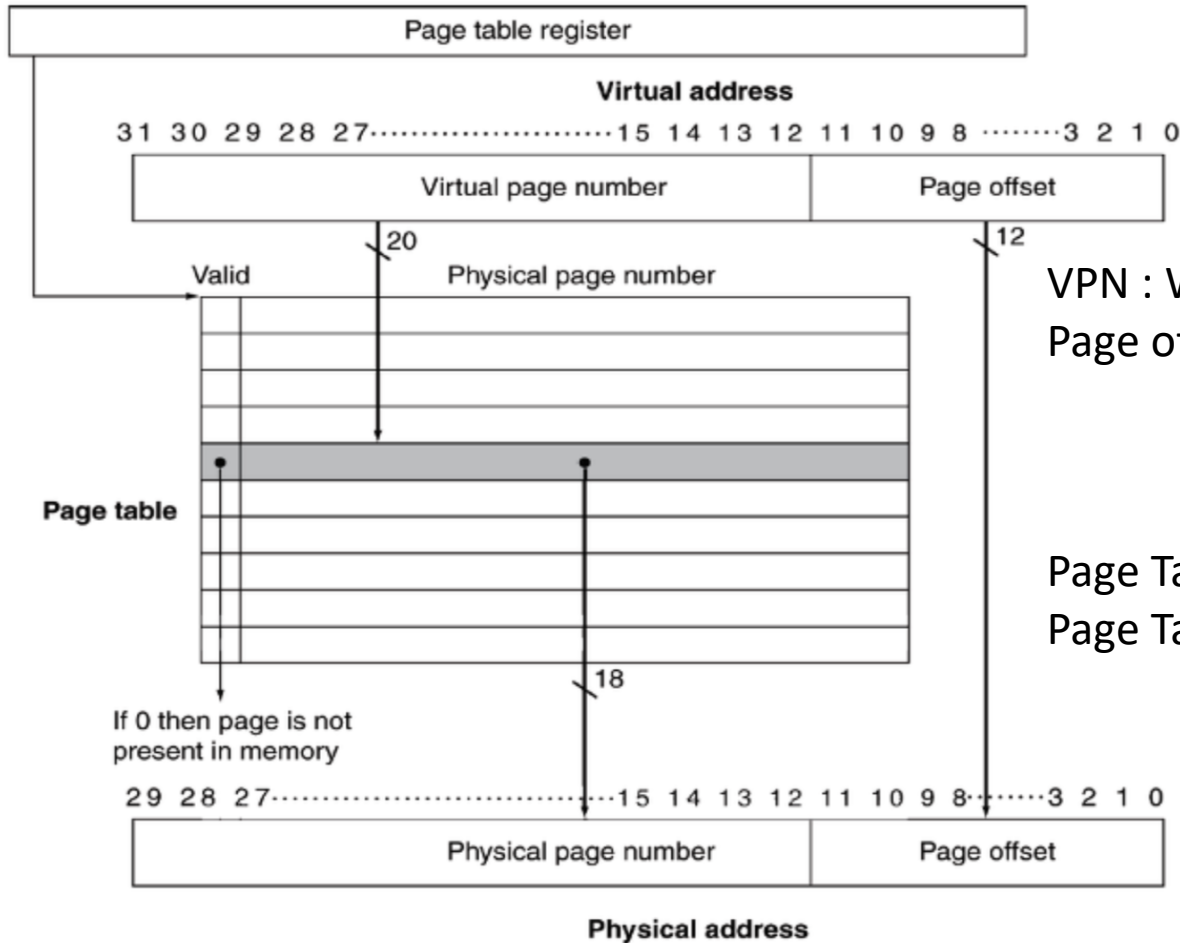
    // Tracing process tree from input_pid to init(1) process

    // Make Output Format string: process_command (process_id)

    return length;
}

static const struct file_operations dbfs_fops = {    Begin of code <- file write operation
    .write = write_pid_to_input,
};
```

# Part2. Virtual to Physical Address Translation



VPN : Which page to look for

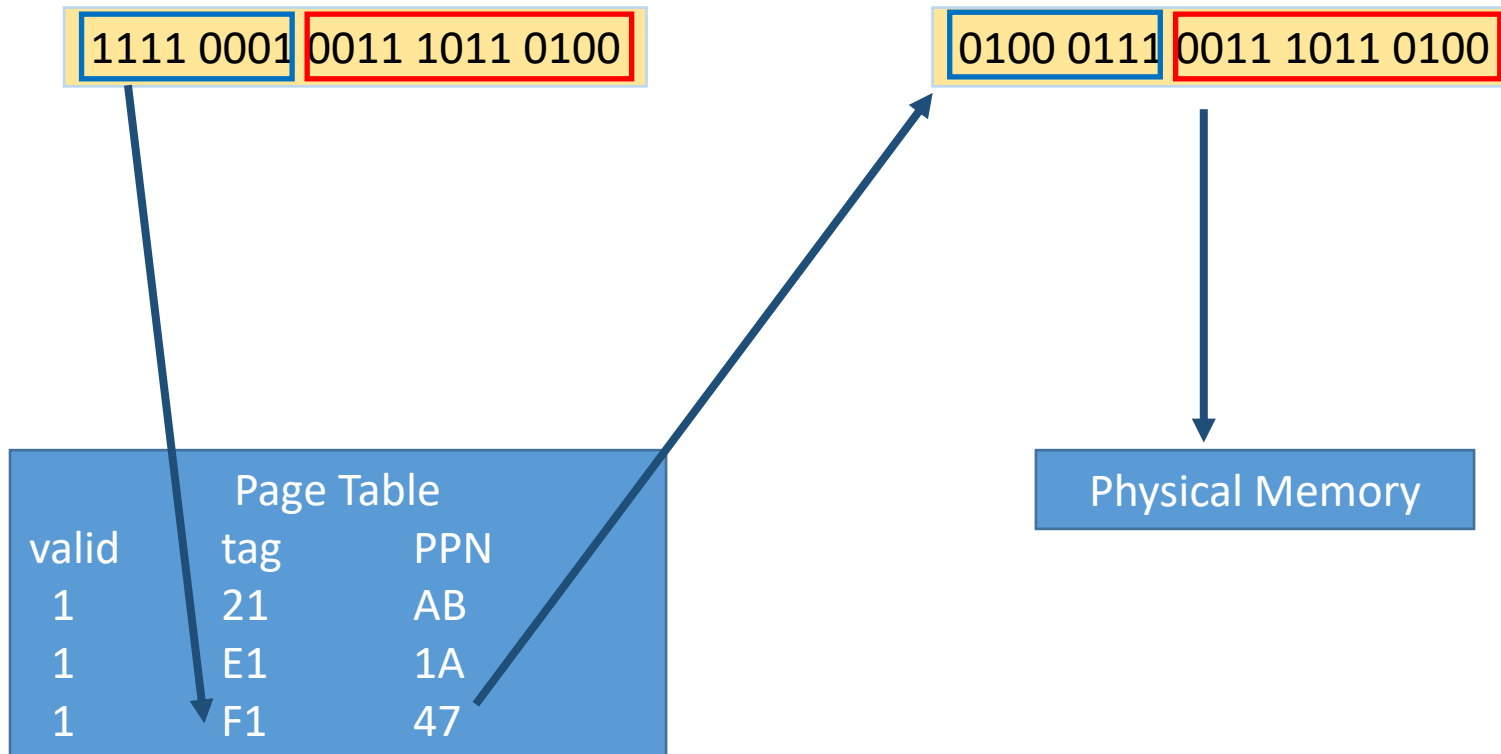
Page offset : Where in the page to look for

Page Table : Contains process page info

Page Table Entry : Record of Page Table

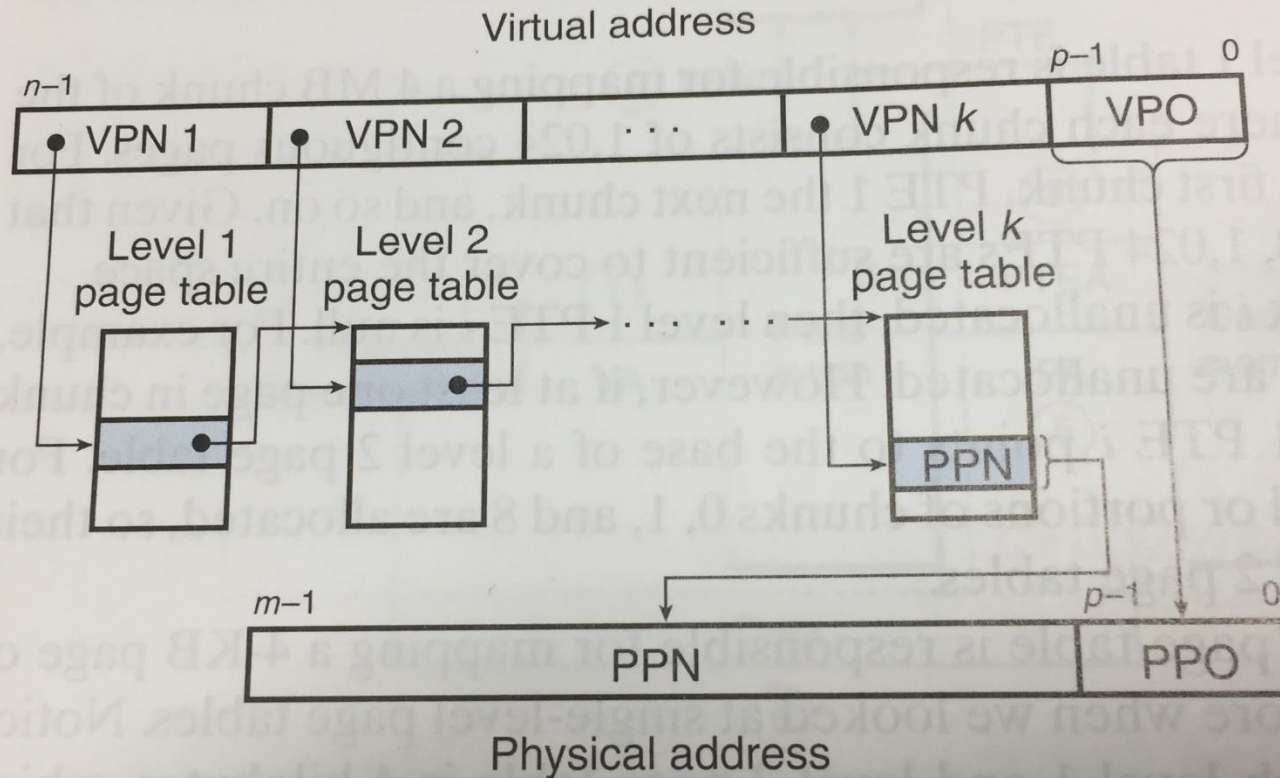
# Example

- Virtual to Physical Translation
  - 4KB page size, 20bit virtual address
  - VA : 0xF13B4





# Multilevel Page Tables



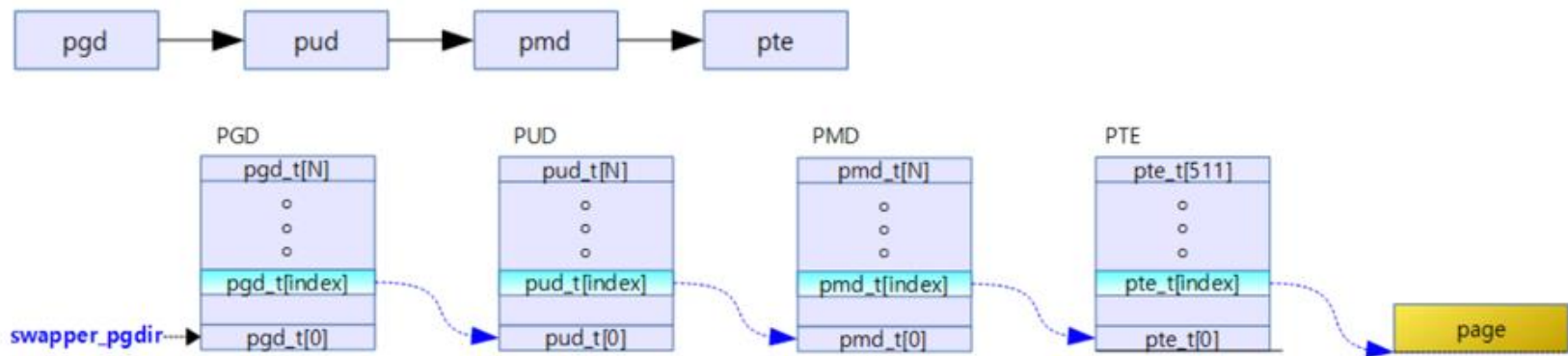
**Figure 9.18** Address translation with a  $k$ -level page table.

# Specification

- In app.c (do not change)
  - makes a virtual address mapped to predefined physical address
  - Compares return value from kernel module with predefined value.
- In your kernel module
  - 1. get pid of app and virtual address
  - 2. find physical address from page table walk
  - 3. returns physical address

# Hints

- Page walk API
  - headers in */usr/src/linux/*
  - *<kernel source>/arch/x86/include/asm/pgtable.h*
  - *<kernel source>/include/linux/pgtable.h*
- Look for the schemes how virtual address is translated to physical address
- Page walk procedure in linux 4.4.0 (5 level in higher ver.)



# Testing your program

- Step
  - 0. `sudo su`
  - 1. `make`
  - 2. `./app`

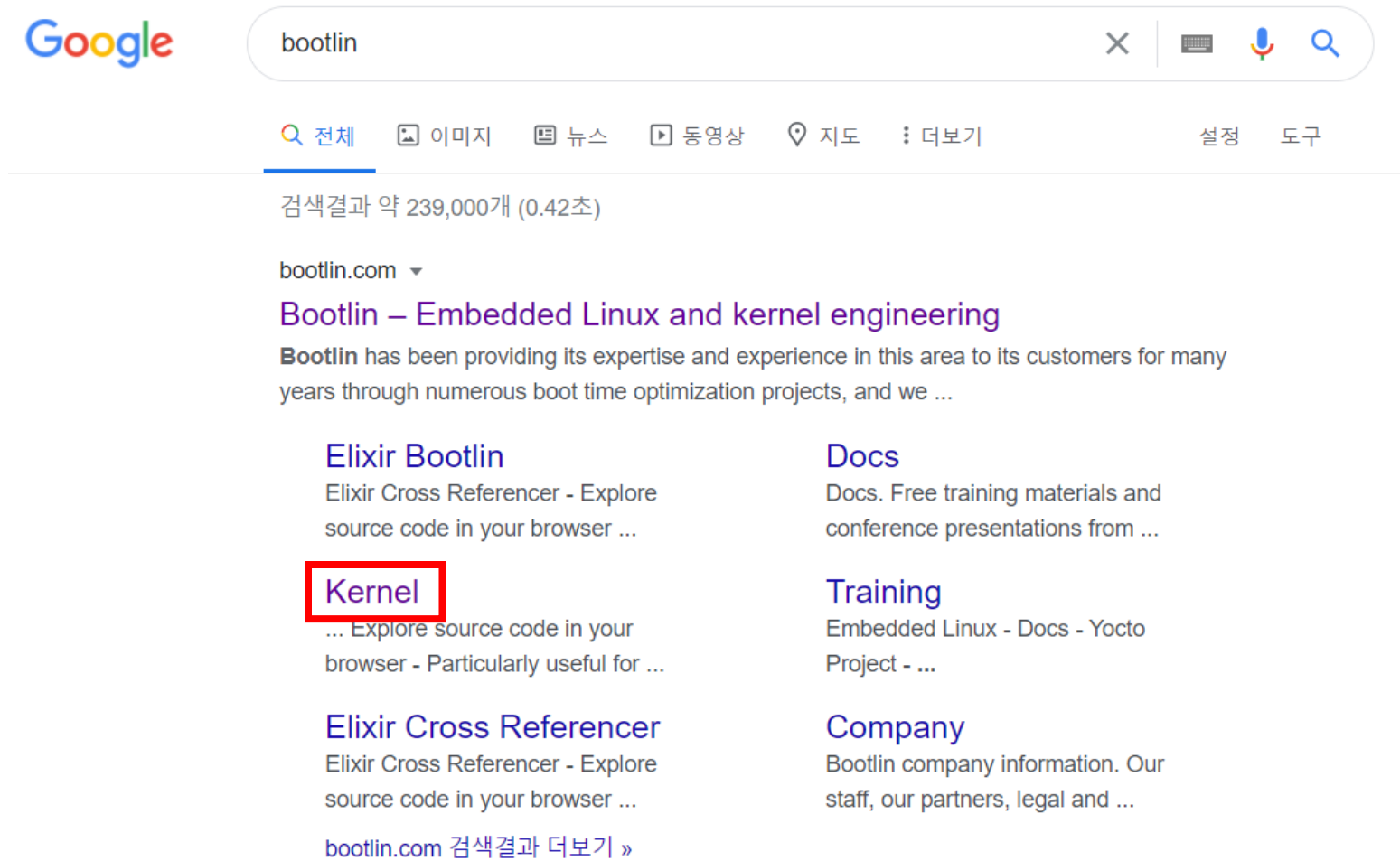
```
root@yschoi-VirtualBox:/home/yschoi/kernellab_full/solution/paddr# sudo su
root@yschoi-VirtualBox:/home/yschoi/kernellab_full/solution/paddr# make
make -C /lib/modules/4.15.0-47-generic/build M=/home/yschoi/kernellab_full/solution/paddr modules;
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-47-generic'
  CC [M]  /home/yschoi/kernellab_full/solution/paddr/dbfs_paddr.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/yschoi/kernellab_full/solution/paddr/dbfs_paddr.mod.o
  LD [M]  /home/yschoi/kernellab_full/solution/paddr/dbfs_paddr.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-47-generic'
gcc -o app app.c;
sudo insmod dbfs_paddr.ko
root@yschoi-VirtualBox:/home/yschoi/kernellab_full/solution/paddr# ./app
vaddr { 7ffbf301000 } paddr { 0 }
vaddr { 7ffbf301000 } paddr { 0 }
vaddr { 7ffbf301000 } paddr { 234512000 }
[TEST CASE]      PASS
root@yschoi-VirtualBox:/home/yschoi/kernellab_full/solution/paddr#
```

# Hints. Helpful kernel functions & data structures

- struct dentry
- struct task\_struct
- struct list\_head
  - INIT\_LIST\_HEAD()
  - list\_add()
  - list\_for\_each\_entry()
- copy\_from\_user()
- copy\_to\_user()
- debugfs\_remove\_recursive()

# Bootlin

- Bootlin([kernel - Linux source code \(v5.10.11\) – Bootlin](#))



The screenshot shows a Google search interface with the query 'bootlin'. The search results page displays the Bootlin website as the top result. The search bar at the top contains the text 'bootlin' and has icons for clearing the search, voice search, and image search. Below the search bar are navigation links for '전체' (All), '이미지' (Images), '뉴스' (News), '동영상' (Videos), '지도' (Maps), and '더보기' (More). The search results section shows 'bootlin.com' as the source, with the title 'Bootlin – Embedded Linux and kernel engineering'. The description states: 'Bootlin has been providing its expertise and experience in this area to its customers for many years through numerous boot time optimization projects, and we ...'. Below the title, there are several links: 'Elixir Bootlin', 'Docs', 'Training', and 'Company'. The 'Kernel' link is highlighted with a red box. At the bottom, there is a link to 'bootlin.com 검색결과 더보기 »'.

Google

bootlin

전체 이미지 뉴스 동영상 지도 더보기

설정 도구

검색결과 약 239,000개 (0.42초)

bootlin.com ▾

**Bootlin – Embedded Linux and kernel engineering**

Bootlin has been providing its expertise and experience in this area to its customers for many years through numerous boot time optimization projects, and we ...

**Elixir Bootlin**  
Elixir Cross Referencer - Explore source code in your browser ...

**Kernel**  
... Explore source code in your browser - Particularly useful for ...

**Elixir Cross Referencer**  
Elixir Cross Referencer - Explore source code in your browser ...

**Docs**  
Docs. Free training materials and conference presentations from ...

**Training**  
Embedded Linux - Docs - Yocto Project - ...

**Company**  
Bootlin company information. Our staff, our partners, legal and ...

bootlin.com 검색결과 더보기 »

# Kernel Debugging

---

- dmesg
  - dmesg -w
- printk

# (extra) Tools for kernel programming

---

- Some useful tools in kernel programming
  - tmux
  - ctags
  - cscope



# tmux (terminal multiplexer)

- Installation
  - `sudo apt install tmux`
- Basic Commands

| Command                                        | Description                           |
|------------------------------------------------|---------------------------------------|
| <code>tmux</code>                              | start tmux                            |
| <code>tmux new -s &lt;name&gt;</code>          | start tmux with <name>                |
| <code>tmux ls</code>                           | shows the list of sessions            |
| <code>tmux a #</code>                          | attach the detached-session           |
| <code>tmux a -t &lt;name&gt;</code>            | attach the detached-session to <name> |
| <code>tmux kill-session -t &lt;name&gt;</code> | kill the session <name>               |
| <code>tmux kill-server</code>                  | kill the tmux server                  |

# Split windows with tmux

- **1. ctrl + b (or a)** to type tmux command
- 2. Split vertically : `%(shift 5)`
- 3. Split horizontally : `”(shift ‘)`
- <https://tmuxguide.readthedocs.io/en/latest/tmux/tmux.html>

# Ctags

- What is Ctags?
  - A Tool that makes it easy to navigate big source code projects.
- Ctags generates **database** of tag file
  - for global variables, functions, macros, etc
  - to point where they are declared & defined
- Installation
  - `sudo apt-get install ctags (or exuberant-ctags)`
- Check
  - `ctags --version`
- Help
  - `Ctags --help`

# Ctags – how to make tags file

- Steps
  - 1. go to root directory of codes you want to navigate.
    - `cd /(where your root directory of code is)`
  - 2. generate tags file
    - type `ctags -R` (recursive)
    - or `ctags file1, file2, ...`
  - 3. Check tags file
    - `ls`

```
1. ta@sp3: ~/yschoi/malloclab/src (ssh)
ta@sp3:~/yschoi/malloclab/src$ ctags -R
ta@sp3:~/yschoi/malloclab/src$ ls
checkalign  clock.o  fcyc.o  ftimer.c  Makefile-handout  memlib.c  mm-explicit.c  mm.o  README-handout
checkalign.c  config.h  fsecs.c  ftimer.h  mdriver          memlib.h  mm.h          mm-test.c  tags
clock.c      fcyc.c   fsecs.h  ftimer.o  mdriver.c        memlib.o  mm-implicit.c  mm-tree.c
clock.h      fcyc.h   fsecs.o  Makefile  mdriver.o        mm.c      mm-naive.c    README
```

- 4. Remove tags file
  - `rm tags`

# Ctags – how to use

- Case 1. In code file
  - 1. place cursor on the keyword you want to locate where it is defined
  - 2. type **ctrl + ]**

```
227 if ((bp = mem_sbrk(size)) == (void *)-1)
228     return NULL;
229
230 /* Initialize free block header/footer and the epilogue header */
231 PUT(HDRP(bp), PACK(size, 0)); /* free block header */
232 PUT(FTRP(bp), PACK(size, 0)); /* free block footer */
233 PUT(HDRP(NEXT_BLKPB(bp)), PACK(0, 1)); /* new epilogue header */
234
235 /* Coalesce if the previous block was free */
236 return coalesce(bp);
237 }
238 /* $end mmextendheap */
239
240 /*
241  * place - Place block of asize bytes at start of free block bp
242  *         and split if remainder would be at least minimum block size
243  */
244 /* $begin mmplace */
245 /* $begin mmplace-proto */
246 static void place(void *bp, size_t asize)
247 {
248     /* ... */
249 }
```

227:17 [62%]  
"mm.c" 369L, 9640C

```
58 void *mem_sbrk(int incr)
59 {
60     char *old_brk = mem_brk;
61
62     if ( (incr < 0) || ((mem_brk + incr) > mem_max_addr) ) {
63         errno = ENOMEM;
64         fprintf(stderr, "ERROR: mem_sbrk failed. Ran out of memory...\n");
65         return (void *)-1;
66     }
67     mem_brk += incr;
68     return (void *)old_brk;
69 }
70
71 /*
72  * mem_heap_lo - return address of the first heap byte
73  */
74 void *mem_heap_lo()
75 {
76     return (void *)mem_start_brk;
77 }
```

58:1 [61%]  
'memlib.c' 101L, 2270C

- 3. type **ctrl + t** to go back

# Ctags – how to use

- Case 2. In tags file
  - 1. **vi tags**
  - 2. type **:tj [tag name]** to find

```
34 FREE_MASK mm-tree.c 65;" d file:
35 FTRP mm-implicit.c 71;" d file:
36 FTRP mm.c 71;" d file:
37 GET mm-implicit.c 62;" d file:
38 GET mm.c 62;" d file:
1:1 [Top]
:tj mem_sbrk
```

- 3. type **:po** to comeback

```
58 void *mem_sbrk(int incr)
59 {
60     char *old_brk = mem_brk;
61
62     if ( (incr < 0) || ((mem_brk + incr) > mem_max_addr)) {
63         errno = ENOMEM;
64         fprintf(stderr, "ERROR: mem_sbrk failed. Ran out of memory...\n");
65         return (void *)-1;
66     }
67     mem_brk += incr;
68     return (void *)old_brk;
69 }
70
71 /*
72  * mem_heap_lo - return address of the first heap byte
73  */
74 void *mem_heap_lo()
75 {
76     return (void *)mem_start_brk;
77 }
58:1 [61%]
:po
```

# Ctags – how to use

---

- Case 3. Vim?
  - 구글 선생님께 여쭙볼 것.

# Using Ctags in Linux kernel code

```

!_TAG_FILE_FORMAT      2           /extended format; --format=1 will
not append ";" to lines/
!_TAG_FILE_SORTED      1           /0=unsorted, 1=sorted, 2=foldcase/
!_TAG_PROGRAM_AUTHOR    Darren Hiebert /dhiebert@users.sourceforge
e.net/
!_TAG_PROGRAM_NAME      Exuberant Ctags //
!_TAG_PROGRAM_URL       http://ctags.sourceforge.net /official
site/
!_TAG_PROGRAM_VERSION   5.9~svn20110310 //
$0      arch/mips/include/asm/mach-cavium-octeon/kernel-entry-init
.h      /^      dins      v0, $0, 0, 6$/"      v
$0      arch/mips/include/asm/mach-cavium-octeon/kernel-entry-init
.h      /^      sd      $0, -32768(v0)$/"      v
$0      arch/mips/include/asm/stackframe.h      /^
NG_S    $0, PT_R0(sp)$/"      v
$0      arch/mips/include/asm/stackframe.h      /^
r      v1, $0, v1$/"      v
$1      arch/mips/include/asm/asmmacro.h      /^      _cfcmsa $1
, MSA_CSR$/"      v
$1      arch/mips/include/asm/asmmacro.h      /^      sw      $1
, THREAD_MSA_CSR(\\thread)$/"      v
$10     arch/mips/include/asm/stackframe.h      /^
i_ld    $10, PT_R10, \\docfi$/"      v
$10     arch/mips/include/asm/stackframe.h      /^
i_st    $10, PT_R10, \\docfi$/"      v
$11     arch/mips/include/asm/stackframe.h      /^
i_ld    $11, PT_R11, \\docfi$/"      v
$11     arch/mips/include/asm/stackframe.h      /^
i_st    $11, PT_R11, \\docfi$/"      v
$12     arch/mips/include/asm/stackframe.h      /^
i_ld    $12, PT_R12, \\docfi$/"      v
$12     arch/mips/include/asm/stackframe.h      /^
i_st    $12, PT_R12, \\docfi$/"      v
$13     arch/mips/include/asm/stackframe.h      /^
i_ld    $13, PT_R13, \\docfi$/"      v
$13     arch/mips/include/asm/stackframe.h      /^
i_st    $13, PT_R13, \\docfi$/"      v
$14     arch/mips/include/asm/stackframe.h      /^
i_ld    $14, PT_R14, \\docfi$/"      v
$14     arch/mips/include/asm/stackframe.h      /^
i_st    $14, PT_R14, \\docfi$/"      v
:tj task_struct

```

```

#include <linux/debugfs.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/uaccess.h>
#include <linux/list.h>
#include <linux/slab.h>

#define COMM_STR_SIZE    32
#define BLOB_SIZE        128

MODULE_LICENSE("GPL");

static struct dentry *dir, *inputdir, *ptreedir;
static struct task_struct *curr;

LO
no
struct task_list {
    struct list_head list;
    pid_t pid;
    char comm[COMM_STR_SIZE];
};

cf struct task_list t_list;

cf struct debugfs_blob_wrapper p_tree;
char blob[BLOB_SIZE];

cf
static void add_task(struct task_struct *task)
{
    struct task_list *node;

    node = (struct task_list*)kmalloc(sizeof(struct task_lis
    t), GFP_KERNEL);
    node->pid = task->pid;
    strncpy(node->comm, task->comm, COMM_STR_SIZE);

    list_add((struct list_head*)node, &(t_list.list));
}

cf
static void build_blob(void)
{
    char buffer[BLOB_SIZE] = "";

    "dbfs_ptree.c" 104 lines, 2481 characters

```



# Using Ctags in Linux kernel code

```
        u8                exp_need_qs;

        /* Otherwise the compiler can store garbage here: */
        u8                pad;
    } b; /* Bits. */
    u32 s; /* Set of bits. */
};

enum perf_event_task_context {
    perf_invalid_context = -1,
    perf_hw_context = 0,
    perf_sw_context,
    perf_nr_task_contexts,
};

struct wake_q_node {
    struct wake_q_node *next;
};

struct task_struct {
#ifdef CONFIG_THREAD_INFO_IN_TASK
    /*
     * For reasons of header soup (see current_thread_info()),
     * this
     * must be the first element of task_struct.
     */
    struct thread_info        thread_info;
#endif
    /* -1 unrunnable, 0 runnable, >0 stopped: */
    volatile long             state;

    /*
     * This begins the randomizable portion of task_struct. On
     * ly
     * scheduling-critical items should be added above here.
     */
    randomized_struct_fields_start

    void                    *stack;
    atomic_t                usage;
};

#include <linux/debugfs.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/uaccess.h>
#include <linux/list.h>
#include <linux/slab.h>

#define COMM_STR_SIZE      32
#define BLOB_SIZE          128

MODULE_LICENSE("GPL");

static struct dentry *dir, *inputdir, *ptreedir;
static struct task_struct *curr;

struct task_list {
    struct list_head list;
    pid_t pid;
    char comm[COMM_STR_SIZE];
};

struct task_list t_list;

struct debugfs_blob_wrapper p_tree;
char blob[BLOB_SIZE];

static void add_task(struct task_struct *task)
{
    struct task_list *node;

    node = (struct task_list*)kmalloc(sizeof(struct task_list), GFP_KERNEL);
    node->pid = task->pid;
    strncpy(node->comm, task->comm, COMM_STR_SIZE);

    list_add((struct list_head*)node, &(t_list.list));
}

static void build_blob(void)
{
    char buffer[BLOB_SIZE] = "";

    "dbfs ptree.c" 104 lines, 2481 characters
}
```

# Cscope

- A tool to navigate in big source code.
- Diff with Ctags?
  - Able to locate **functions where they are called** too.
- Installation
  - `sudo apt-get install cscope`
- Check
  - `cscope --version`

```
ta@sp3:~/yschoi/malloclab/src$ cscope --version
cscope: version 15.8b
ta@sp3:~/yschoi/malloclab/src$
```

- Help
  - `cscope --help`

# Cscope – how to make cscope database file

- Steps
  - 1. go to root directory of codes you want to navigate.
    - `cd /(where your root directory of code is)`
  - 2. generate cscope database file
    - `find ./ -name '*[cCsShH]' > file_list`
    - `cscope -i file_list`
  - 3. Check cscope.out file
    - `ls`

```
ta@sp3:~/yschoi/malloclab/src$ find ./ -name '*[cCsShH]' > file_list
ta@sp3:~/yschoi/malloclab/src$ cscope -i file_list
ta@sp3:~/yschoi/malloclab/src$ ls
checkalign  cscope.out  fsecs.h    Makefile-handout  memlib.o    mm.o
checkalign.c  fcyc.c      fsecs.o    mdriver           mm.c         mm-test.c
clock.c       fcyc.h      ftimer.c   mdriver.c         mm-explicit.c mm-tree.c
clock.h       fcyc.o      ftimer.h   mdriver.o         mm.h         README
clock.o       file_list   ftimer.o   memlib.c          mm-implicit.c README-handout
config.h      fsecs.c     Makefile   memlib.h          mm-naive.c
```

- 4. Remove tags file
  - `rm cscope.out file_list`

# Cscope – how to use

- 1. type **cscope -d** to execute
- 2. type **ctrl+d** to break out

Cscope version 15.8b

Press the ? key for help

Find this C symbol:  
Find this global definition:  
Find functions called by this function:  
Find functions calling this function: **mem\_sbrk**  
Find this text string:  
Change this text string:  
Find this egrep pattern:  
Find this file:  
Find files #including this file:  
Find assignments to this symbol:

Functions calling this function: mem\_sbrk

| File            | Function         | Line                                                               |
|-----------------|------------------|--------------------------------------------------------------------|
| mm-explicit.c   | requestMoreSpace | 172 ptrNewBlock = (void *)((unsigned int )mem_sbrk(totalSize - 4); |
| 1 mm-explicit.c | mm_init          | 204 if (mem_sbrk(initsize) == (void *)-1) {                        |
| 2 mm-implicit.c | mm_init          | 99 if ((heap_listp = mem_sbrk(4*WSIZE)) == NULL)                   |
| 3 mm-implicit.c | extend_heap      | 227 if ((bp = mem_sbrk(size)) == (void *)-1)                       |
| 4 mm-naive.c    | mm_malloc        | 62 void *p = mem_sbrk(newsize);                                    |
| 5 mm-test.c     | mm_init          | 43 mem_sbrk(64000);                                                |
| 6 mm-test.c     | mm_malloc        | 54 void *p = mem_sbrk(newsize);                                    |
| 7 mm-tree.c     | mm_init          | 726 if (mem_sbrk(HEAP_INITSIZE) == NULL)                           |
| 8 mm-tree.c     | mm_realloc       | 826 if (mem_sbrk(grow_size) == NULL)                               |
| 9 mm-tree.c     | mm_malloc        | 875 if (mem_sbrk(block_size) == NULL)                              |

Find this C symbol:  
Find this global definition:  
Find functions called by this function:  
Find functions calling this function:  
Find this text string:  
Change this text string:  
Find this egrep pattern:  
Find this file:  
Find files #including this file:  
Find assignments to this symbol:

# FAQ part1

- (1) init process의 이름이 systemd
  - 상관 없습니다.
- (2) Corner cases?
  - 채점 기준: 다음 명령이 올바르게 작동 하는지 확인, 모든 pid는 valid
    - # make
    - # cd /sys/kernel/debug/ptree
    - # echo pid1 >> input
    - # cat ptree
    - # echo pid2 >> input
    - # cat ptree
    - ...
    - # echo pidn >> input
    - # cat ptree
    - # cd -
    - # make clean

# FAQ part2

- 1. mmap fails in app
  - Try using different PADDRs
  - PADDR must be 4KB-aligned (ex. 0x1234**000**)
  - Please report the value used to TA
- 2. Accessing user buffer data in kernel
  - Recommended: use `copy_from_user()`
- 3. Flushing kernel data to user
  - Recommended: use `copy_to_user()`

# Next Time

---

- Questions
  - eTL Q&A Board
- Please, read the handout & start early!
- Next time (May. 16<sup>th</sup>)
  - **Kernel LAB Q&A session in this class room**

# References

---

- Linux Kernel Module Programming Guide
  - <http://www.tldp.org/LDP/lkmpg/2.6/html/>
- Debugfs APIs
  - <https://www.kernel.org/doc/Documentation/filesystems/debugfs.txt>
- Makefile Guide
  - <https://www.cs.duke.edu/~ola/courses/programming/Makefiles/Makefiles.html>



# References

- Tmux guide
  - <https://tmuxguide.readthedocs.io/en/latest/tmux/tmux.html>
- Ctags
  - <https://bowbowbow.tistory.com/15>
- Cscope
  - <https://harryp.tistory.com/131>
- Address Translation, Multilevel Page table
  - P.849~855, R. E. Briant, D. R. O'Hallaron, Computer Systems, A programmer's perspective 3<sup>rd</sup> edition