System Programming
Lab #8

2019-04-30

sp-tas

Lab Assignment #4 : Kernel Lab

- Download skeleton code & pdf from eTL
  - kernellab-handout.tar, kernellab-handout.pdf
- Hand In #1 – Setup (Done)
  - capture your development environment
  - Upload your capture image eTL
    - 압축파일 양식 : [학번]_이름]_kernellab_setup.tar (or .zip etc) **(including below files)**
      - filename for part #1 : [학번]_[이름]_kernellab_ptree.jpg (or .png, etc)
      - filename for part #2: [학번]_[이름]_kernellab_paddr.jpg (or .png, etc)
- Hand In #2 – Your Implementation
  - Upload your files eTL
    - 압축파일 양식 : [학번]_[이름]_kernellab.tar (or .zip, etc)
    - Ex) 2017-12345_홍길동_kernellab.tar
  - A zip file should include
  - (1) a tarball of your implementation directory (2) report
    - tarball 양식 : kernellab-[학번].tar.gz eg) kernellab-2019-12345.tar.gz
    - Report 양식 : [학번]_[이름]_kernellab_report.pdf (or .hwp, .txt etc)
- Please, READ the Hand-out and Lab material thoroughly!

Lab Assignment #4 : Kernel Lab

- Step 1. Setup (Done)
  - (part #0) Load my own kernel module
- Step 2. Implementation
  - (part #1) Tracing process tree from process id
  - (part #2) Finding physical address using virtual address

- Assigned : April 23
- Deadline for Step 1. Setup : April 29, 23:59:59 (Delay NOT allowed)
- Deadline for Step 2. Implementation : May 13, 23:59:59
- Delay policy : Same as before

- Lab sessions will be
  - 4/23: Kernel lab part #0, #1
  - 4/30: Kernel lab part #2 ← TODAY
  - 5/7 : Kernel lab Q&A session

Today's Lab

- Some useful tools in kernel programming
  - tmux, ctags, cscope
- Part#2. Finding physical address using virtual address
  - Remind translation of VA->PA in Computer Architecture
  - Assignment spec
  - How to begin
  - Testing your program
- Evaluation

tmux (terminal multiplexer)

- Installation
  - sudo apt-get install tmux

- Basic Commands

Command	Description
tmux	start tmux
tmux new -s <name>	start tmux with <name>
tmux ls	shows the list of sessions
tmux a #	attach the detached-session
tmux a -t <name>	attach the detached-session to <name>
tmux kill-session –t <name>	kill the session <name>
tmux kill-server	kill the tmux server

Split windows with tmux

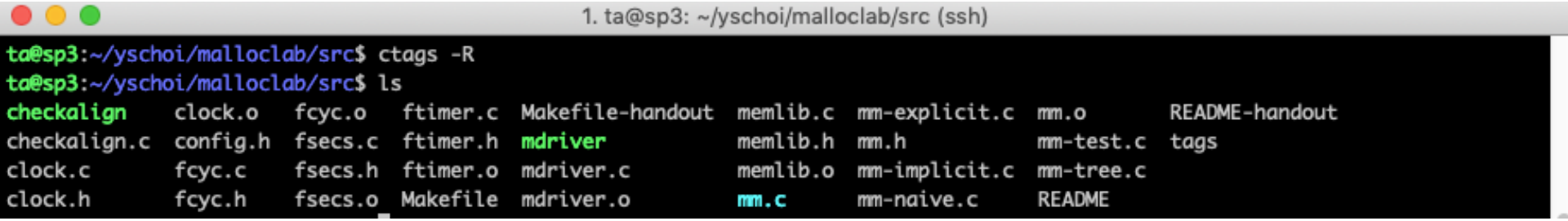
- **1. ctrl + b (or a)** to type tmux command
- 2. Split vertically : %(shift 5)
- 3.Split horizontally : ″(shift ‘)
- https://tmuxguide.readthedocs.io/en/latest/tmux/tmux.html

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



  - 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_BLKP(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)
 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

```
 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
```

Where is linux kernel code?

- /usr/src/linux-headers-4.xx/

```
root@yschoi-VirtualBox:/usr/src/linux-headers-4.15.0-47# ls
arch            drivers   ipc       Makefile  scripts   tools
block           firmware  Kbuild    mm        security  ubuntu
certs           fs        Kconfig   net       sound     usr
crypto          include   kernel    new.c     spl       virt
Documentation   init      lib       samples   tags      zfs
root@yschoi-VirtualBox:/usr/src/linux-headers-4.15.0-47# vi tags
```

```
#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_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));
}

static void build_blob(void)
{
        char buffer[BLOB_SIZE] = "";
"dbfs_ptree.c" 104 lines, 2481 characters
```

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.sourceforg
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      /^              LO
NG_S  $0, PT_R0(sp)$/;"       v
$0      arch/mips/include/asm/stackframe.h      /^              no
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      /^              cf
i_ld  $10, PT_R10, \\docfi$/;"        v
$10     arch/mips/include/asm/stackframe.h      /^              cf
i_st  $10, PT_R10, \\docfi$/;"        v
$11     arch/mips/include/asm/stackframe.h      /^              cf
i_ld  $11, PT_R11, \\docfi$/;"        v
$11     arch/mips/include/asm/stackframe.h      /^              cf
i_st  $11, PT_R11, \\docfi$/;"        v
$12     arch/mips/include/asm/stackframe.h      /^              cf
i_ld  $12, PT_R12, \\docfi$/;"        v
$12     arch/mips/include/asm/stackframe.h      /^              cf
i_st  $12, PT_R12, \\docfi$/;"        v
$13     arch/mips/include/asm/stackframe.h      /^              cf
i_ld  $13, PT_R13, \\docfi$/;"        v
$13     arch/mips/include/asm/stackframe.h      /^              cf
i_st  $13, PT_R13, \\docfi$/;"        v
$14     arch/mips/include/asm/stackframe.h      /^              cf
i_ld  $14, PT_R14, \\docfi$/;"        v
$14     arch/mips/include/asm/stackframe.h      /^              cf
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;

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_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));
}

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;
:po
```

```
#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_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));
}

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

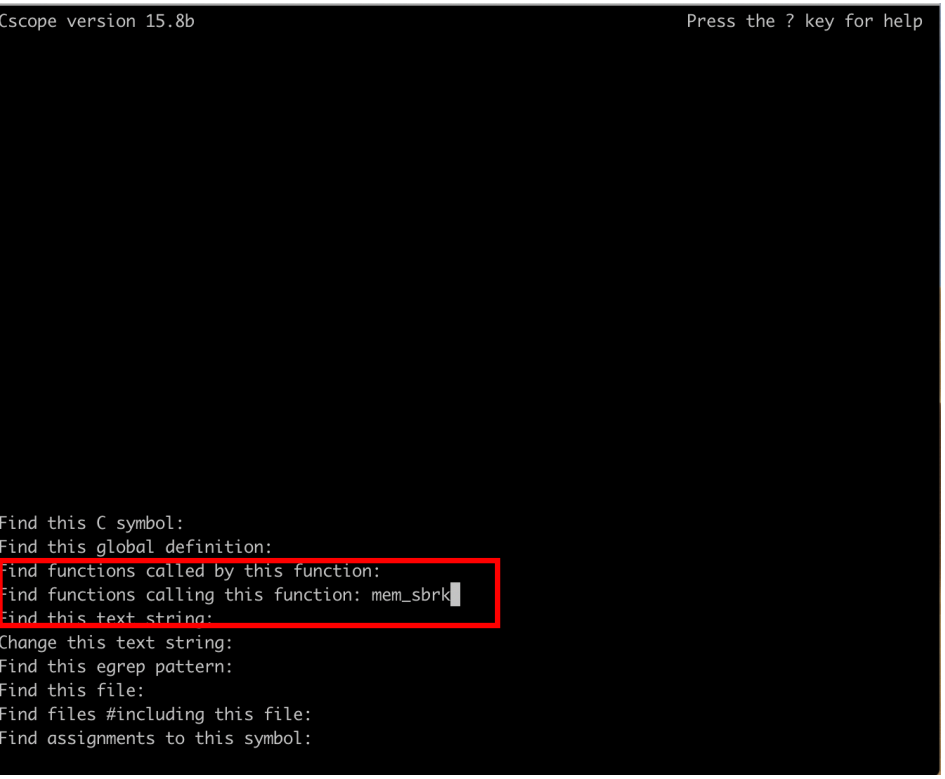
- Steps
  - 1. go to root directory of codes you want to navigate.
    - cd /(where your root directory of code is)
  - 2. generate cscope database file
    - find ./ -name '*[cCsShH]]' > file_list
    - cscope -i file_list
  - 3. Check cscope.out file
    - ls

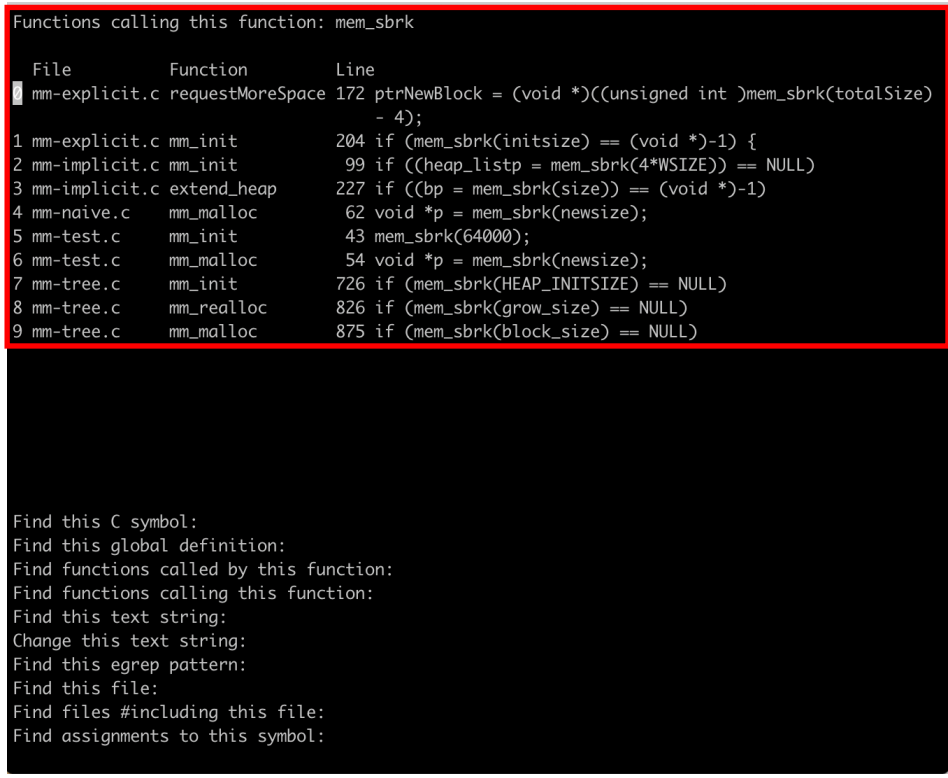
```
ta@sp3:~/yschoi/malloclab/src$ find ./ -name '*[cCsShH]' > file_list
ta@sp3:~/yschoi/malloclab/src$ cscope -i file_list
ta@sp3:~/yschoi/malloclab/src$ ls
checkalign    cscope.out  fsecs.h   Makefile-handout  memlib.o       mm.o
checkalign.c  fcyc.c      fsecs.o   mdriver           mm.c          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

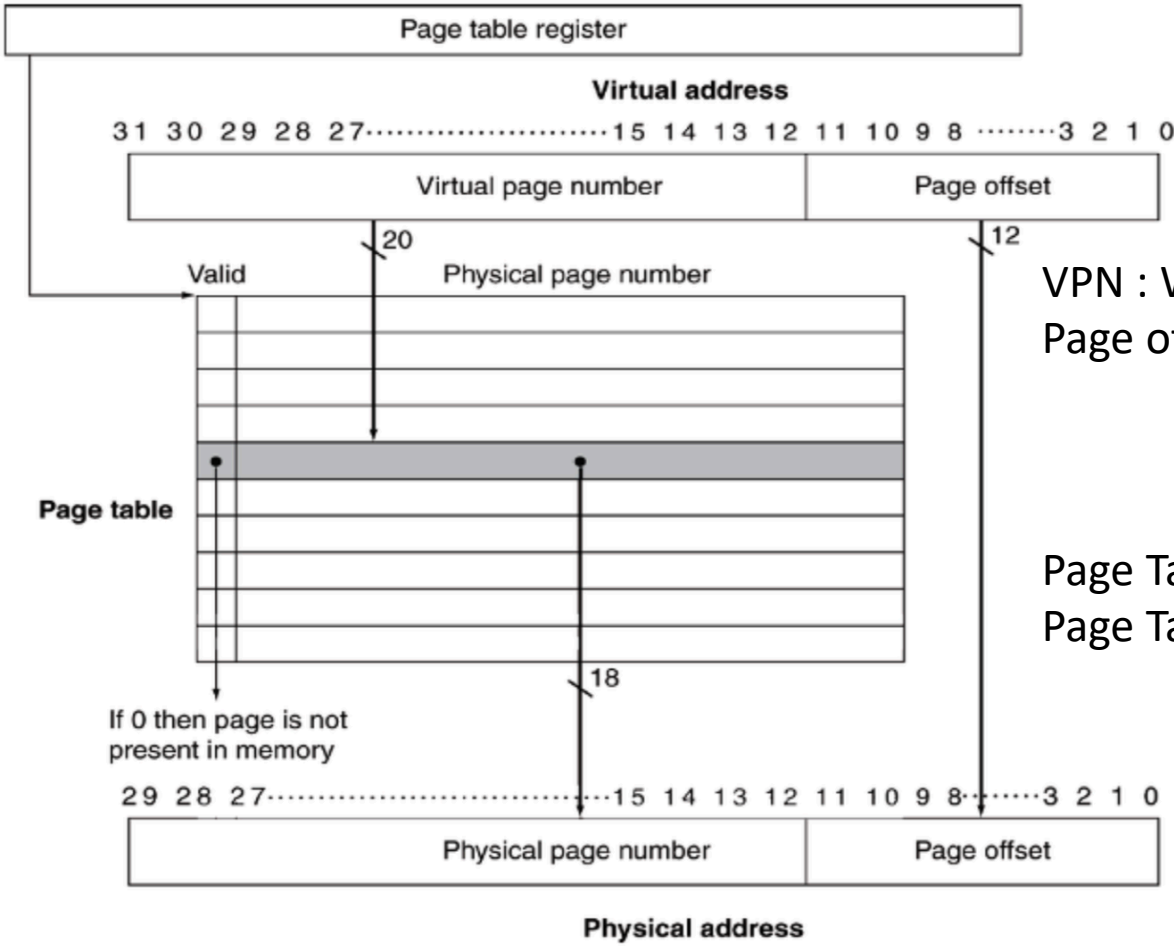
Cscope – how to use

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





Virtual to Physical Address Translation

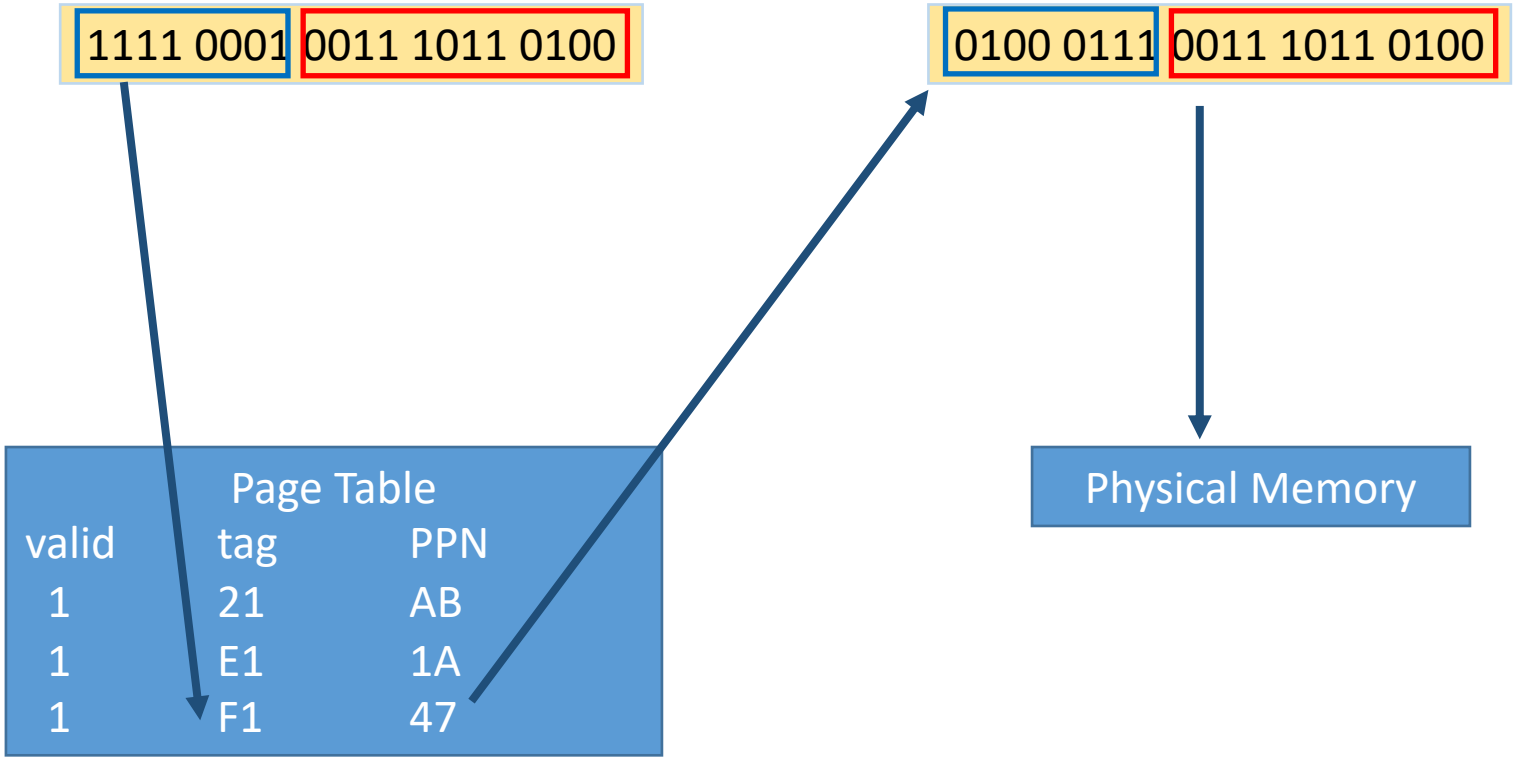


VPN : Which page to look for
Page offset : Where in the page to look for

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

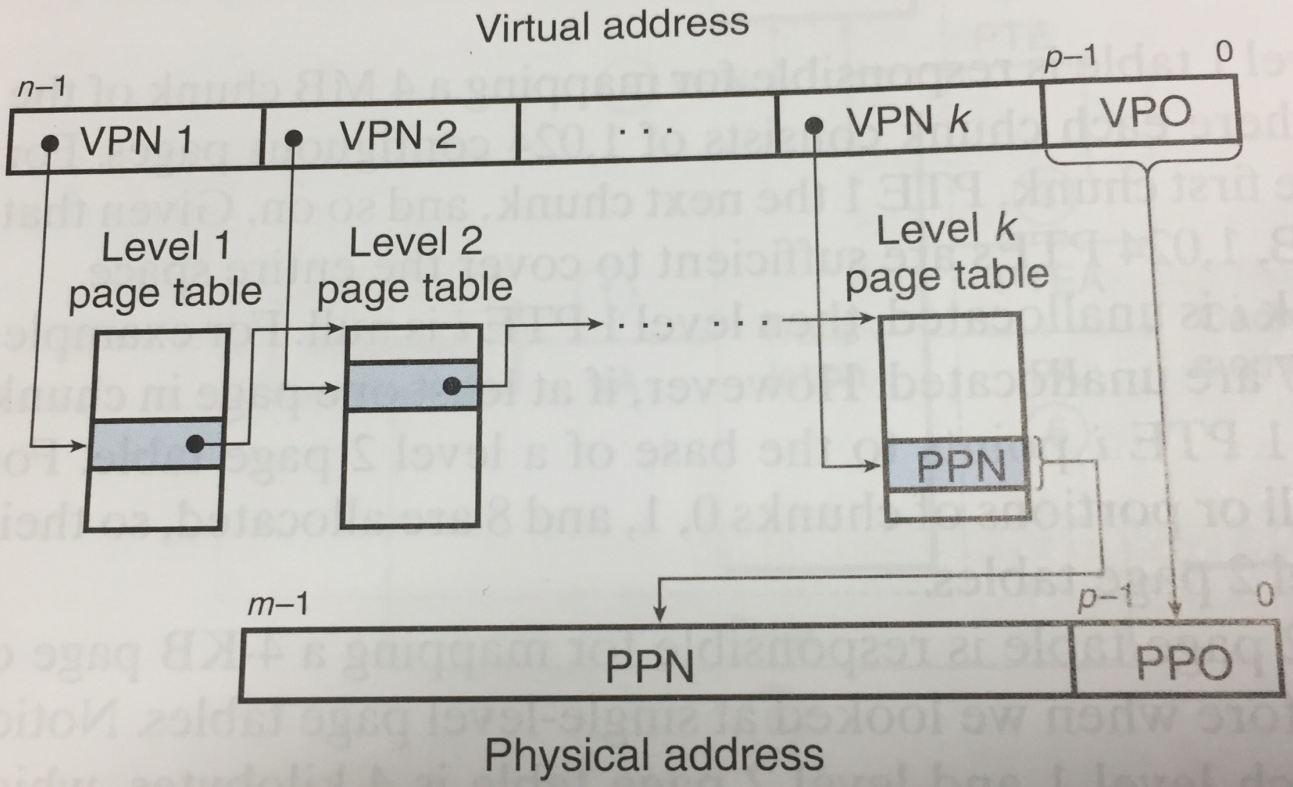


Figure 9.18 Address translation with a k-level page table.

Part2. Finding physical address using virtual address

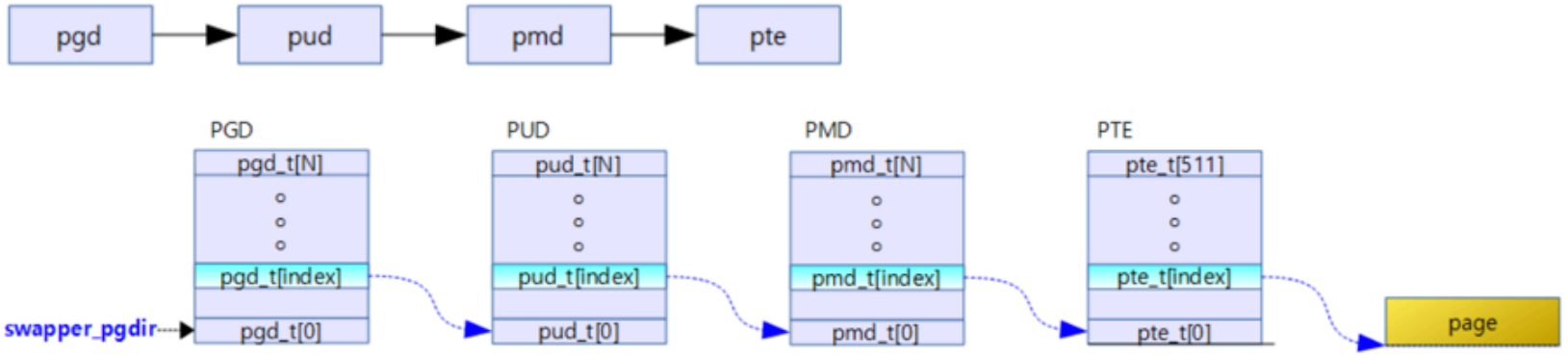
- Spec
  - In app.c
    - makes a virtual address mapped to predefined physical address
  - In your kernel module
    - 1. get pid of app and virtual address
    - 2. returns physical address
  - In app.c
    - Compares return value from kernel module with predefined value.

Hints

- Page walk API
  - /usr/src/linux/arch/x86/include/asm/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/solut
ion/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 { 7ffbfe301000 } paddr { 0 }
vaddr { 7ffbfe301000 } paddr { 0 }
vaddr { 7ffbfe301000 } paddr { 234512000 }
[TEST CASE]    PASS
root@yschoi-VirtualBox:/home/yschoi/kernellab_full/solution/paddr#
```

Asked Questions

- 0. setup
  - (1) 사전에 공지해드린 대로 delay는 받지 않습니다.
- 1. ptree
  - (1) init process의 이름이 systemd
    - -> 상관 없습니다.
  - (2) kernel message 출력 형식
    - -> 형식은 자유롭게 하셔도 됩니다.
    - 정상 동작하는 상황에서 출력되는 메시지는 제출 시 주석 처리.
    - Error case에서 출력되는 메시지는 놔두셔도 상관 없습니다.
      - -> 테스트할 때 출력되면 안되겠죠?
- 2.paddr
  - (1) huge page(2M~256M)의 case
    - -> 고려하지 않아도 됩니다.
  - (2) test input case
    - -> invalid vaddr은 test하지 않습니다.

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 3rd edition