System Programming
Lab #8

2020-05-13

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) 2020-12345_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-2020-12345.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 : May 6
- Deadline for Step 1. Setup : May 13, 23:59:59 (Delay NOT allowed)
- Deadline for Step 2. Implementation : May 27, 23:59:59
- Delay policy : Same as before

- Lab sessions will be
  - 5/6: Kernel lab part #0, #1
  - 5/13: Kernel lab part #2 ← TODAY
  - 5/27 : Kernel lab Q&A session

Notice

- Please re-download the handout .tar and .pdf file
  - Makefile bug fixed
  - Source code and description for part #2 updated

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

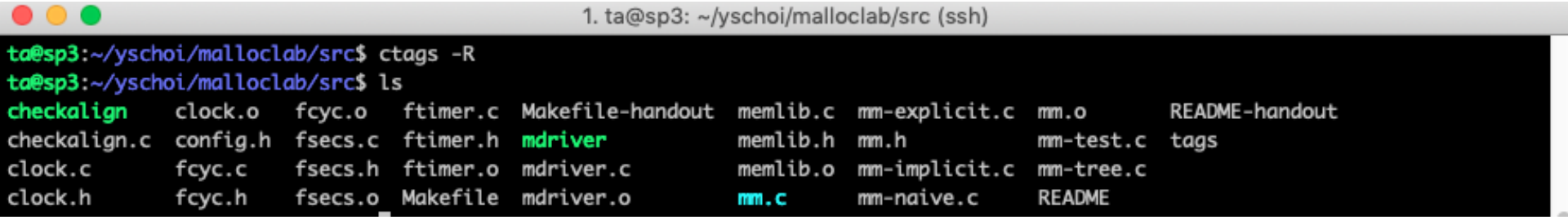
- **1. ctrl + b (or a)** to type tmux command
- 2. Split vertically : %(shift 5)
- 3.Split horizontally : ″(shift ‘)
- https://tmuxguide.readthedocs.io/en/latest/tmux/tmux.html

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
```

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

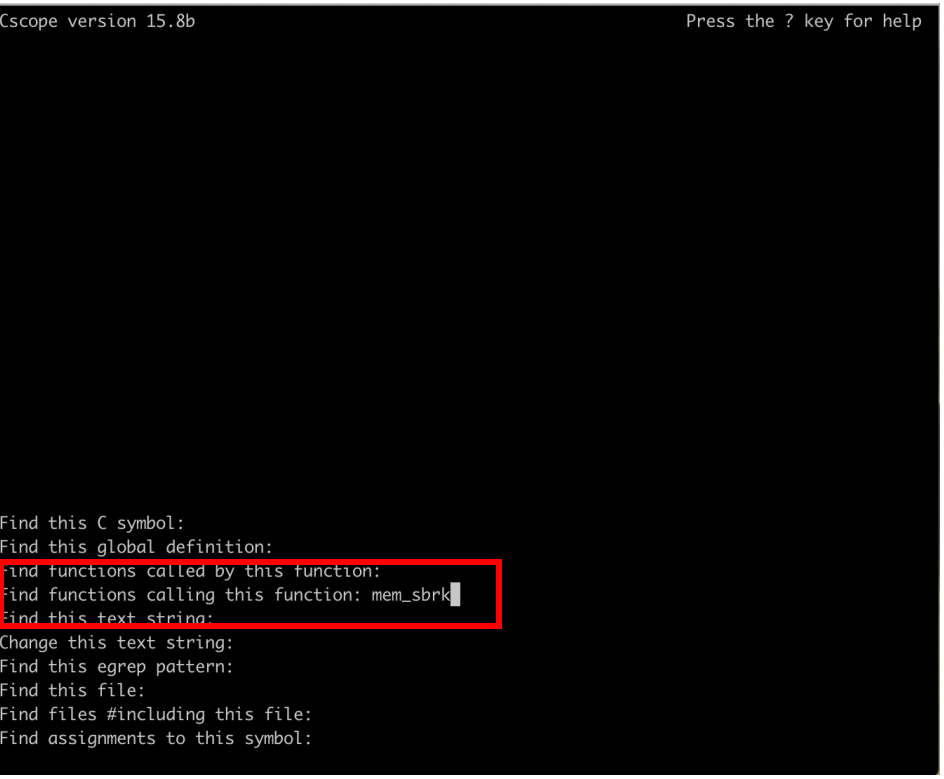
- Steps
  - 1. go to root directory of codes you want to navigate.
    - cd /(where your root directory of code is)
  - 2. generate cscope database file
    - find ./ -name '*[cCsShH]]' > file_list
    - cscope -i file_list
  - 3. Check cscope.out file
    - ls

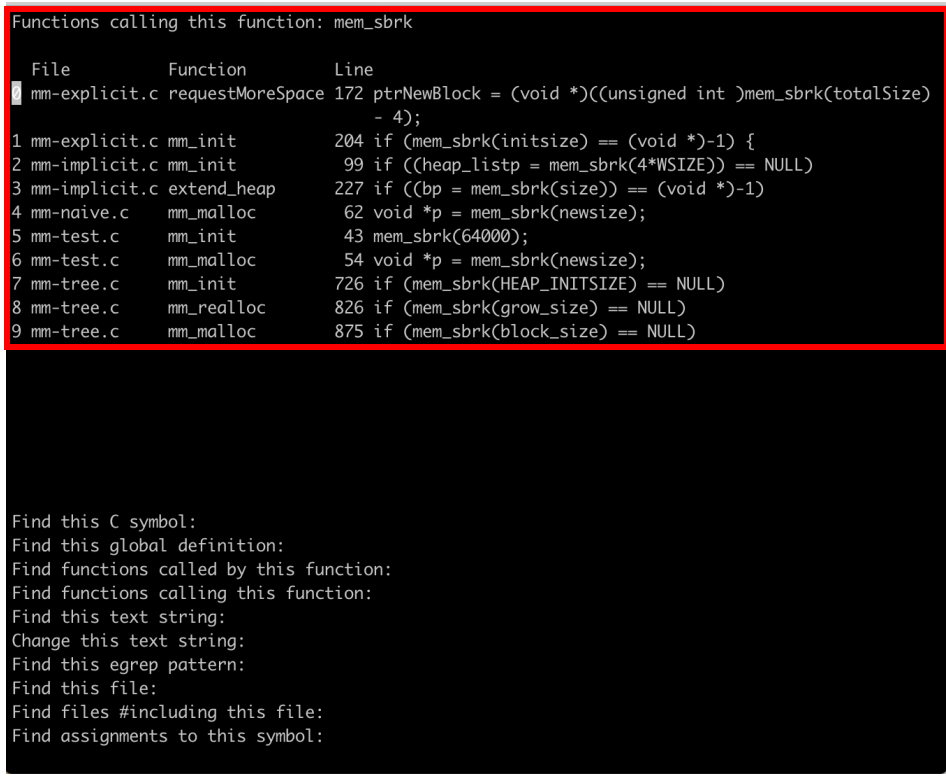
```
ta@sp3:~/yschoi/malloclab/src$ find ./ -name '*[cCsShH]' > file_list
ta@sp3:~/yschoi/malloclab/src$ cscope -i file_list
ta@sp3:~/yschoi/malloclab/src$ ls
checkalign    cscope.out  fsecs.h   Makefile-handout  memlib.o       mm.o
checkalign.c  fcyc.c      fsecs.o   mdriver           mm.c          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





Downloading Linux kernel source code

- https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.15.tar.gz
- You can download the source code using wget
- Usage: $ wget <source>

```
ubuntu@sp:~/src$ wget https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.15.tar.gz
--2020-05-12 14:59:30--  https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.15.tar.gz
Resolving mirrors.edge.kernel.org (mirrors.edge.kernel.org)... 147.75.95.133, 2604:1380:3000:1500::1
Connecting to mirrors.edge.kernel.org (mirrors.edge.kernel.org)|147.75.95.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 157656459 (150M) [application/x-gzip]
Saving to: 'linux-4.15.tar.gz'

linux-4.15.tar.gz                  19%[===========>                                        ]  28.93M  2.53MB/s    eta 50s
```

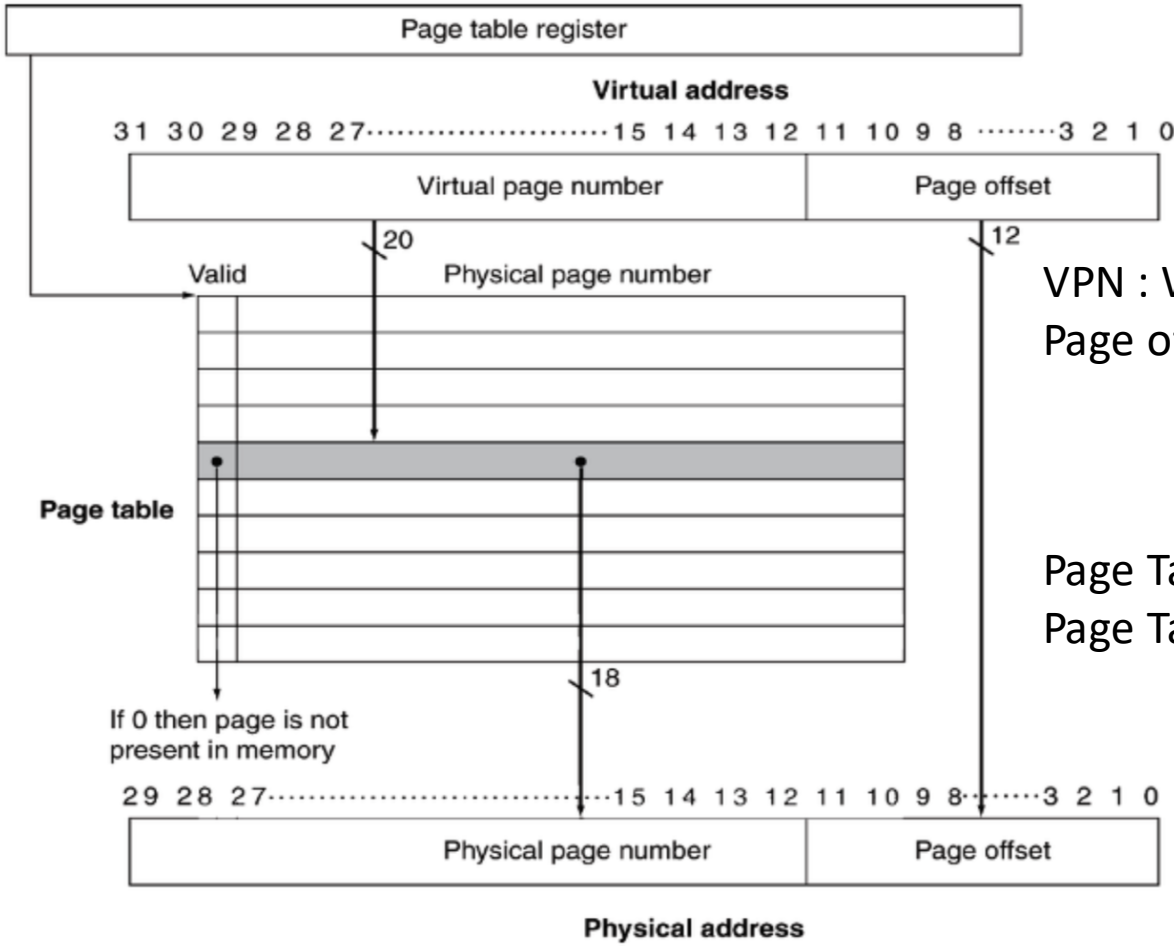
Playing with Linux kernel source code

- Uncompress the tar archive
  - $ tar –xzvf linux-4.15.tar.gz
- Go to the root of the source code
  - $ cd linux-4.15/
- Create tag files using Linux kernel's makefile
  - $ make tags cscope ARCH=x86_64 –j2
- Now we are all set!
  - Example: $ vi –t debugfs_create_file

Playing with Linux kernel source code

- $ vi –t <symbol>
- Inside vim
  - g + ctrl +]
  - :tj
  - ctrl + t
- $ cscope

Virtual to Physical Address Translation

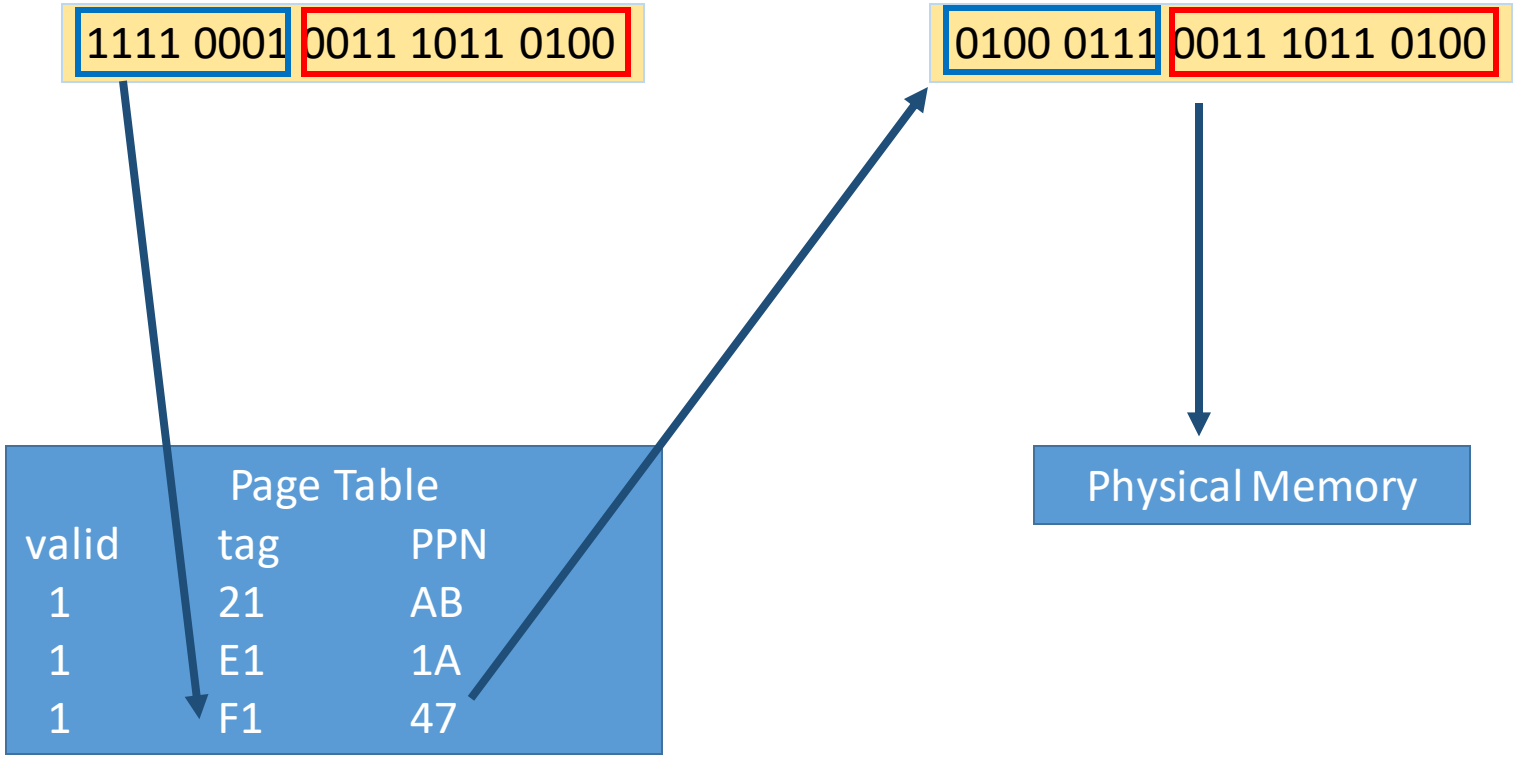


VPN : Which page to look for
Page offset : Where in the page to look for

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

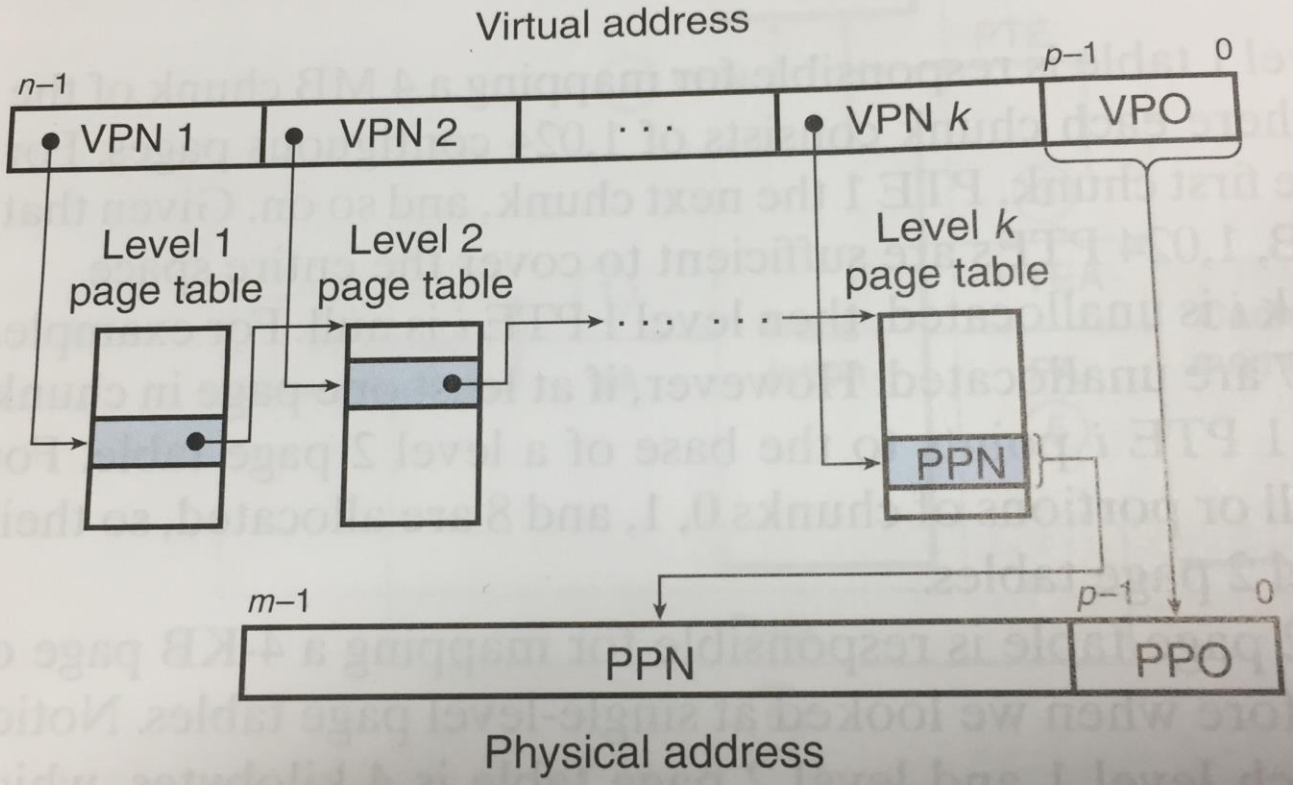


Figure 9.18 Address translation with a k-level page table.

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
  - <Linux source root>/arch/x86/include/asm/pgtable.h
- Look for the schemes how virtual address is translated to physical address
- Page walk procedure in Linux 4.15.0
  - pgd -> p4d -> pud -> pmd -> pte

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

- 1. ptree
  - (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

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