Project #0: Installing Pintos

[CSE4070]

Instructor

Prof. Youngjae Kim

Teaching Assistants

Junhyeok Park, Sungjin Byeon (01)
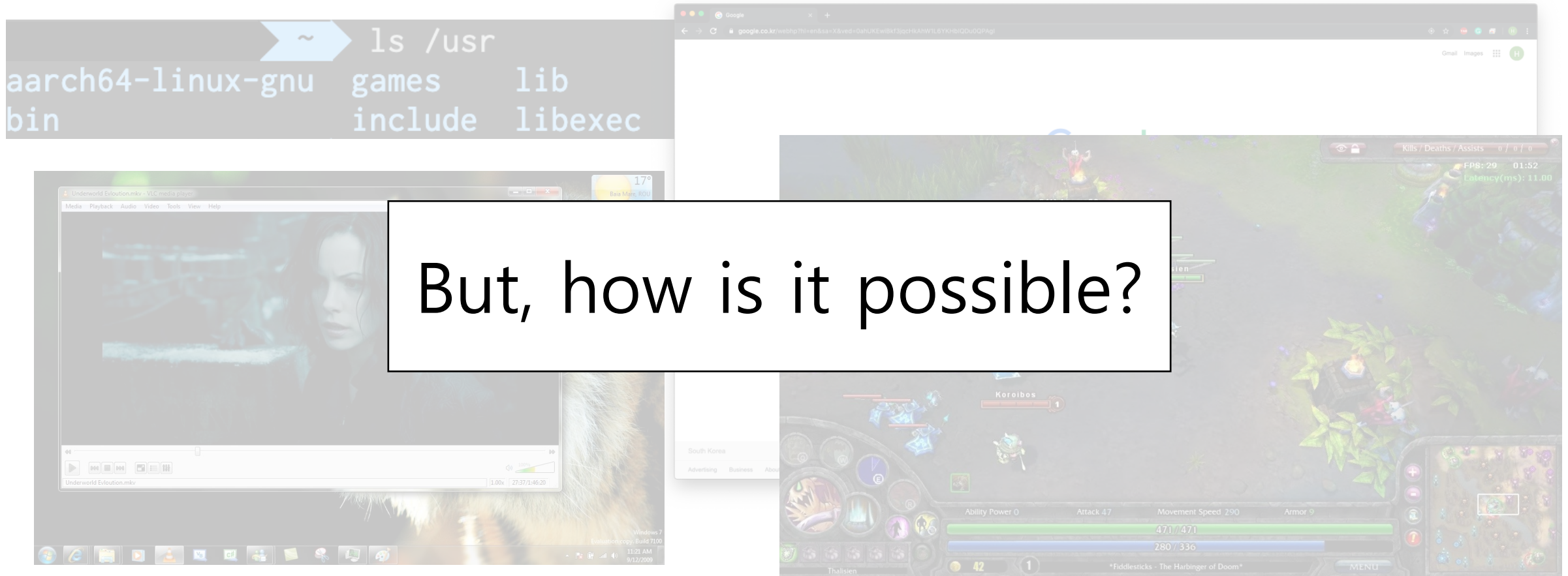
Junghyun Ryu, Seoyeong Lee (02)

Fall 2024

서강대학교
SOGANG UNIVERSITY

Introduction to PintOS

Operating Systems

- We usually double click icon to launch applications on Windows
or type commands on Linux and use the applications or see the result.

But, how is it possible?

Operating Systems

- What do we need to run applications?

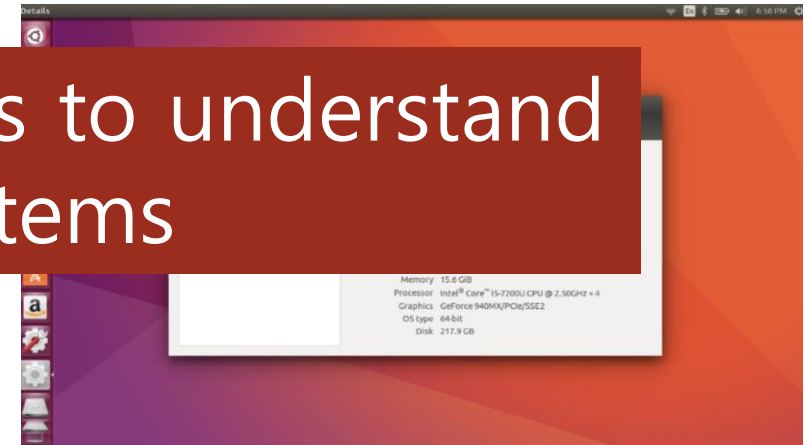- We need some intermediary between hardware and applications.

- It is operating system.

Windows   macOS   Linux

In this class, we will do Pintos projects to understand principles of operating systems
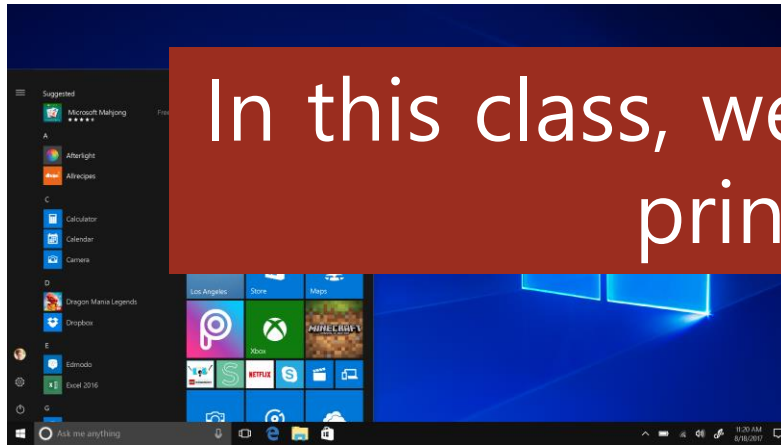
Pintos & Emulator

- Pintos is simple OS framework for 80x86 architecture.

- Use system simulator that simulates an 80x86 CPU and its peripheral devices.

- Project Category : User Programs, Kernel Threads, Virtual Memory, File Systems.

- Features
    1) Support user and kernel thread.
    2) Allow running user program (basic UNIX commands like echo, ls, cat, pwd, …).
    3) Support simple file system.
    4) Implemented in C language.
    5) Well-Documented Project & Grading System.
- We will use QEMU as an emulator for Pintos.

서강대학교
SOGANG UNIVERSITY

Pintos & Emulator

- Both KVM and Qemu are virtualization solutions for the Linux OS. The reason KVM and Qemu are installed together is that KVM and Qemu have a complementary relationship.

- "Virtualization" is to provide functions such as kernel translation and resource distribution through a hypervisor to run an OS that can use the hardware in a virtual machine.

- "Emulation" is the implementation of hardware in software to provide a **specific execution environment.**
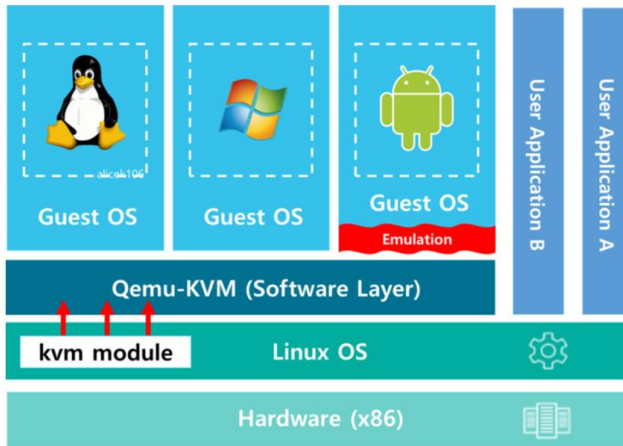
Structures of Pintos
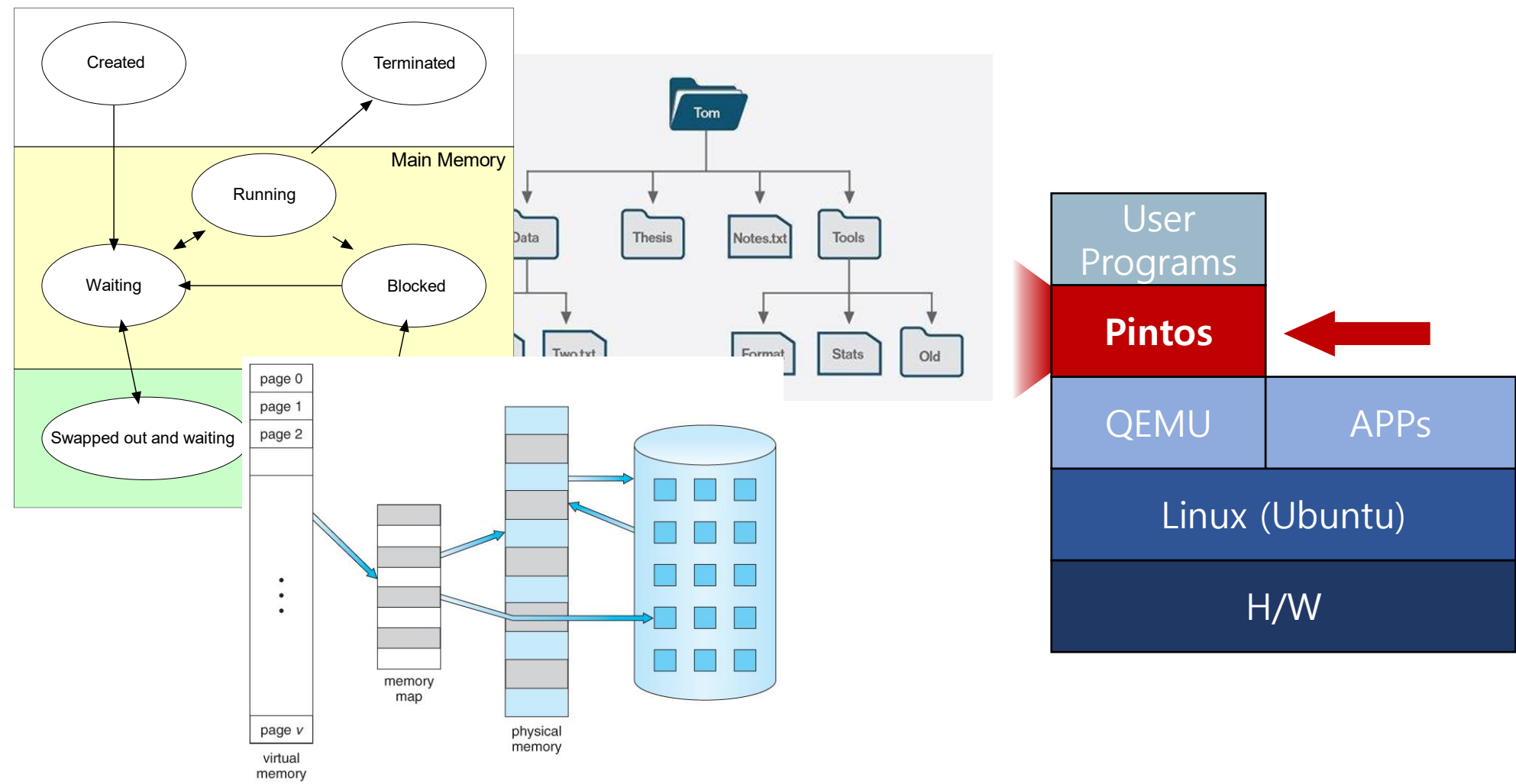
Linux Instructions and Vim Usage

Useful Linux instructions

- man
- mkdir/rmdir
- cp
- mv
- rm
- cat
- echo
- grep
- ps
- kill
- pwd
- su/passwd
- tar

서강대학교
SOGANG UNIVERSITY

man

Provides description and usage for Linux commands

Usage: man [instruction]

Ex:

$ man cp

mkdir/rmdir

Make/remove directory

If the directory is not empty, use 'rm -r' to remove it

Usage: mkdir [option] [Directory Name]

rmdir [option] [Directory Name]

Ex:

$ mkdir temp

cp

Copy the file (Original file is preserved)

Usage: cp [option] [src] [dst]

Ex:

$ cp a.c temp

mv

Move file or rename file (Original file is disappeared)

Usage: mv [option] [src] [dst]

Ex:

$ mv a.c b.c

rm

Remove file or directory.

Usage: rm [option] [filename]

Ex:

$ rm –rf temp

If you perform 'rm -rf *' in /(root) directory, every file will be deleted from the system.

-rf option indicates recursive and force, respectively.

서강대학교
SOGANG UNIVERSITY

cat

1. Print the contents of the file on the standard output.

Usage: cat [option] [filename]

Ex:

$ cat tempfile

$ cat > test.txt   (Get data from standard input; user can input data until user does [Ctrl+D])

$ cat < test.txt   (Print the contents of the file)

2. Concatenate files

Ex:

$ cat test.txt test2.txt > test12.txt (Concatenate test.txt and test2.txt and make a file "test12.txt")

echo

Prints string or system environment variables.

Usage: echo [string...]

Ex:

$ echo $PATH

$ echo x

grep

Print lines matching a pattern from files or standard input.

Usage: grep [option] PATTERN [File…]

-n: print the line and line number in FILE which is matched.

-i: ignore case distinctions.

-l: print only FILE name, which contains PATTERN matched.

Ex:

$ grep –n ftp /etc/groupt

$ grep –i the /etc/init.d/qmail

$ grep –il ftp /etc/init.d/*

ps

Report the list of current processes.

Usage: ps [option]

-ef: print the all processes with full-format listing.

-au: print the username and start time of processes including other users' processes.

Ex:

$ ps -ef

$ ps -au

kill

Sends signal to the processes.

Representative signal is SIGKILL which is used to forcefully terminate a process.

Usage: kill [option] [process id]

-l: print list of signals

Ex:

$ kill -9 4914   (force quit process #4914)

```
sammynam@ubuntu:~/Desktop$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

pwd

Checks the current directory.

Usage: pwd

```
sammynam@ubuntu:~/Desktop$ pwd
/home/sammynam/Desktop
sammynam@ubuntu:~/Desktop$
```

su/passwd

su: switch user ID or become superuser.

passwd: change user password.

Usage: su [options] [username]

Ex:

$ su   (if the USERNAME is omitted than it will switch the account to the superuser.)

$ passwd   (change the password of current account.)

tar

Compresses or extracts file.

Usage: tar [options] [**pathname**]

-c/x: compress / Extract

-v: verbosely list the files processed

-f: use the file to compress or extract

Ex)

$ tar -cvfz sample.tar.gz pintos (if pintos is a directory, it will be compressed into sample.tar.gz)

$ tar -xvfz sample.tar.gz   (extract sample.tar.gz)

서강대학교
SOGANG UNIVERSITY

vim

- Vim is known as Visual Interface iMproved, which is an improved version of vi
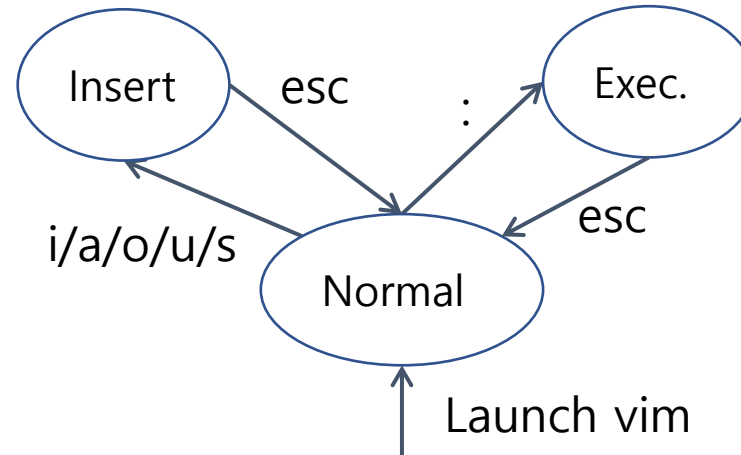
- Normal mode:
  - yy+p: copy and paste line, /: search, x: delete character, dd: delete line, u: undo.
  - v : change into visual mode.
- Insert mode
  - i : Insert, a : Append
- Execution mode (Type ': (colon)' in normal mode)
  - w: save, q: quit, wq!: quit without saving

Data Structures

Data Structures in Pintos Kernel

- Before we dive into Pintos project, we will practice Pintos data structures.

- Pintos provides kernel and user libraries.

- You can find it in "pintos/src/lib/kernel" and "pintos/src/lib/user"

- In this project, we will cover data structures of Pintos kernel libraries.
  → **List, Hash table and Bitmap**

List

- List in Pintos is a **doubly linked list.**
- It is different from usual list structure.
- It splits list element pointers and data.
- struct list_elem
  - Each structure that will be a list item must embed a struct list_elem member.
  - All the list functions operate on list_elem, not the list item.
  - Only list_elem structure is given in the source code.
  - You must implement new structure that consists of list_elem and data.

```
head <-> elem <-> elem <-> elem <-> tail
```

List

- Linked List: Usual way

```
struct list_item
{
    struct list_item *prev
    struct list_item *next
    int data;
}
```
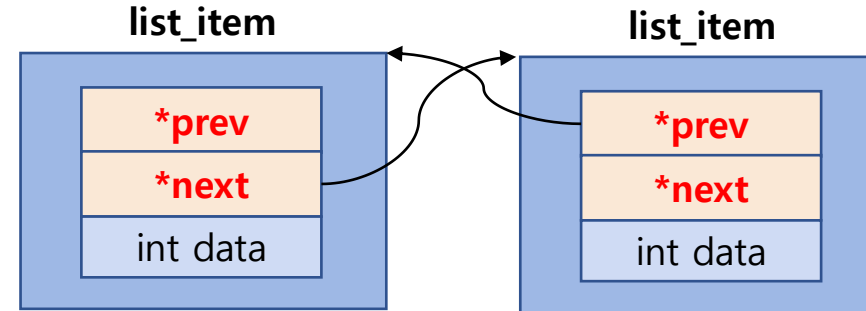

- Linked List: Pintos kernel

```
struct list_elem
{
    struct list_elem *prev;
    struct list_elem *next;
}
```

```
※ struct list_item
{
    struct list_elem elem;
    int data;
    /* Other members you want */
}
```
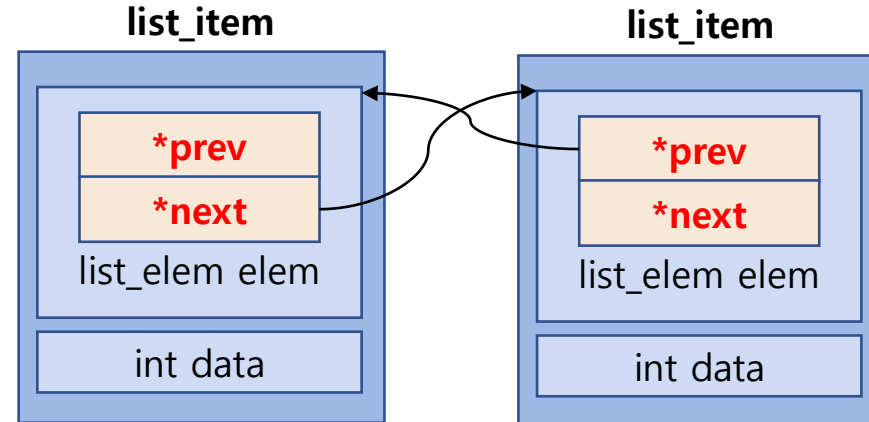

Split the pointer and data

※ 'struct list_item' is not given in the source code, you need to implement it.

List Function Analysis

- **void list_init(struct list *list)**
  - Initializes LIST as an empty list.
  - It should be executed before an element is inserted in LIST.

- **struct list_elem* list_begin(struct list *list)**
  - Returns the first element of LIST.
  - Usually used to iterate the LIST.

- **struct list_elem* list_next(struct list_elem *elem)**
  - Returns the next element of ELEM.
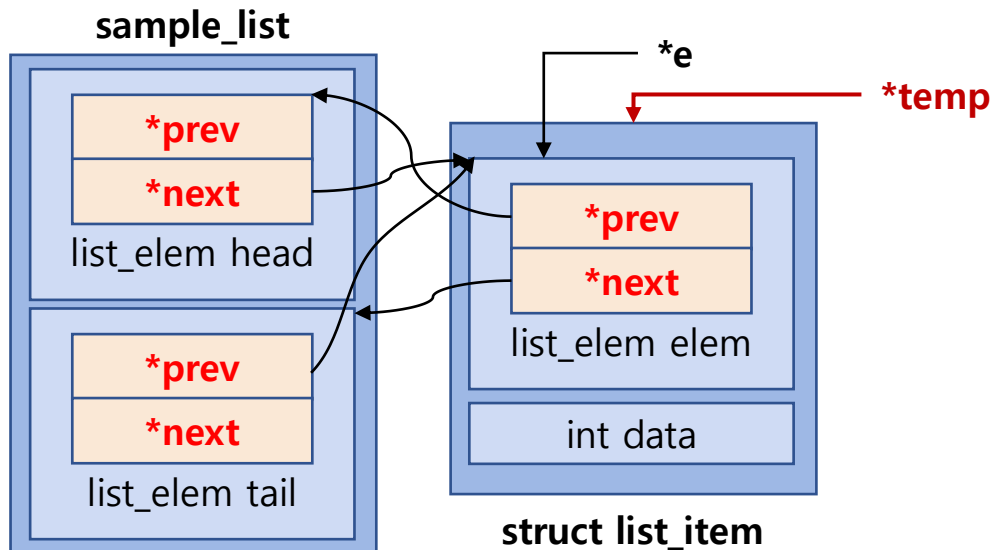 - Usually used to iterate the LIST or search ELEM in the LIST.

List Function Analysis

- **struct list_elem* list_end(struct list *list)**
  - Returns the last ELEM in the LIST.
  - Usually used to iterate the LIST.

- **#define list_entry(list_elem, struct, member)**
  - Converts the pointer to LIST_ELEM into a pointer to STRUCT that LIST_ELEM is embedded inside.
  - Usually used to get address of STURCT which embeds LIST_ELEM.

sample_list

*e

*temp

| *prev |
| *next |

list_elem head

| *prev |
| *next |

list_elem tail

| *prev |
| *next |

list_elem elem

| int data |

struct list_item

/* Assume that there already exists a list, sample_list */
struct list_elem *e;
e = list_begin (&sample_list);
struct list_item *temp = list_entry(e, struct list_item, elem)
int temp_data = temp->data

By using list_elem, we can get address of list_item

Hash Table

- A hash table is a data structure that associates keys with values.
- The primary operation is a lookup.
  - Given a key, find the corresponding value.
- It works by transforming the key using a **hash function** into a hash.

※ **Assume that key and value are same.**

f(x)=x/100

Need to implement.
Refer struct list_item.
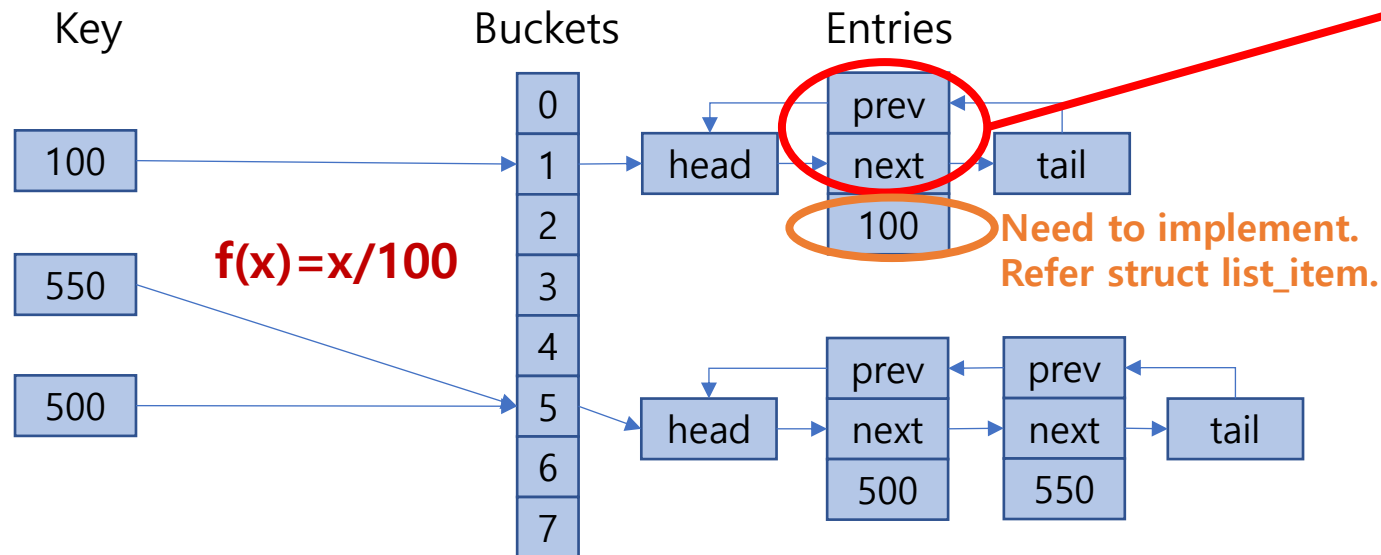
```
struct hash_elem
  {
     struct list_elem list_elem;
  };
```

```
struct hash
  {
     size_t elem_cnt;
     size_t bucket_cnt;
     struct list *buckets;
     hash_hash_func *hash;
     hash_less_func *less;
     void *aux;
  };
```

Hash Table Function Analysis

- **void hash_init(struct hash *h, hash_hash_func *hash, hash_less_func *less, void *aux)**

  - Initializes hash table H and sets <u>hash function HASH</u> and <u>comparison function LESS.</u>

  - You can see the example of hash function such as hash_int, hash_bytes, and hash_string.
    (You have to use hash_int function to pass the test.)

  - Comparison function LESS is used to compare two hash elements.

- **void hash_apply(struct hash *h, hash_action_func *action)**

  - You can apply any ACTION function which you made to hast table H.

  - Used for applying specific function to all elements in hash table.
    e.g.) square function.

  - You can learn the usage of it from 'hash_apply.in' and 'hash_apply.out' in tester directory.

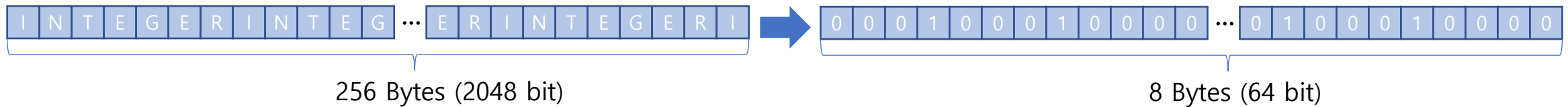# Hash Table Function Analysis

- **#define hash_entry(hash_elem, struct, member)**
  - Converts pointer to HASH_ELEM into a pointer to STRUCT that HASH_ELEM is embedded inside.
  - Usually used to get address of STURCT which embeds HASH_ELEM.

```
struct hash_elem
  {
    struct list_elem list_elem;
  };
```

Bitmap

- A bit array(or bitmap, in some cases) is an array which stores individual bits (Boolean values).
- A bitmap can reduce the waste of memory space.

| I | N | T | E | G | E | R | I | N | T | E | G | ... | E | R | I | N | T | E | G | E | R | I | → | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | ... | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

256 Bytes (2048 bit)　　　　　　　　　　　　　　　　8 Bytes (64 bit)

- Bitmap: Usual way

```
char bitmap[8];
/* Or */
int bitmap[2];
/* Or */
unsigned long bitmap;
```

- Bitmap: Pintos kernel

```
typedef unsigned long elem_type;
struct bitmap
{
    size_t bit_cnt;
    elem_type *bits;
};
```

Bitmap Function Analysis

- **struct bitmap *bitmap_create(size_t bit_cnt)**
  - Initializes a bitmap of BIT_CNT bits and sets all its bits to false.
- **void bitmap_set (struct bitmap *b, size_t idx, bool value)**
  - Atomically sets the bit numbered IDX in B to VALUE.
- **size_t bitmap_count (const struct bitmap *b, size_t start, size_t cnt, bool value)**
  - Returns the number of bits in B between START and START + CNT, exclusive, that are set to VALUE.
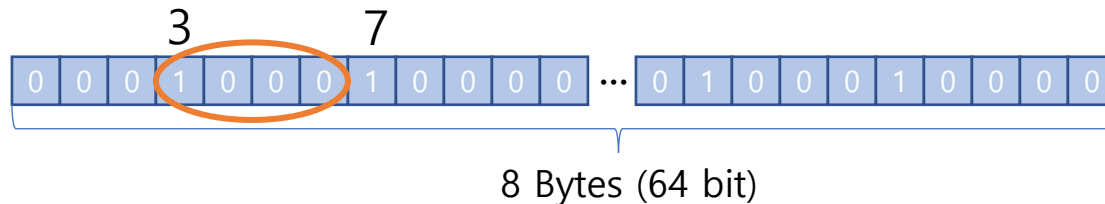
bitmap_count(b, 3, 4, 1) == 1

8 Bytes (64 bit)

Pintos Installation

Caution

- **We will use CSPRO9 (cspro9.sogang.ac.kr) and CSPRO10 (cspro10.sogang.ac.kr)**

- So do not try to run Pintos on CSPRO (cspro.sogang.ac.kr) server

- Note that the CSPRO server indicates CSPRO9 or CSPRO10 from now on

Pintos Installation

1. Download Pintos file

    - We provide modified code in e-class, so don't use original source code from Stanford University.

2. Extract the file

    - $ tar -xvzf pintos_modified.tar.gz

✓ You don't need to install QEMU in the CSPRO sever. It is already installed.

Pintos Installation

- Before running Pintos, we need to setup .bashrc file in the home directory
    1. Open ~/.bashrc with editor.
    2. Add the following line at the end of the file:
       export PATH=/sogang/<u>under</u>/<YOUR_ACCOUNT>/pintos/src/utils:$PATH (학부생)
       export PATH=/sogang/<u>grad</u>/<YOUR_ACCOUNT>/pintos/src/utils:$PATH (대학원생)
    3. Run the following command to apply the changes in bash shell:
       $ source ~/.bashrc

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export PATH=/sogang/under/cse20189999/pintos/src/utils:$PATH
```

Running Pintos

- Build Pintos (assume that you have already extracted the file on your home directory)
    - $ cd ~/pintos/src/threads
    - $ make
    - Consequently, 'build' directory will be created in the current directory (src/threads).

- Run Pintos
    - Pintos provides 'pintos' utility that helps running Pintos by QEMU.
    - 'pintos' utility is in src/utils.
    - **Go to src/threads** and run the following command (you should run it in src/threads, not src/utils).

        ~/pintos/src/threads $ ../utils/pintos -v -- -q run alarm-multiple
        Or
        ~/pintos/src/threads $ pintos -v -- -q run alarm-multiple
        (Note that you should input one space among '-v (turn off VGA)', '--' and '-q (quit after execution')

```
(alarm-multiple) thread 4: duration=50, iteration=6, product=300
(alarm-multiple) thread 4: duration=50, iteration=7, product=350
(alarm-multiple) end
Execution of 'alarm-multiple' complete.
Timer: 580 ticks
Thread: 0 idle ticks, 581 kernel ticks, 0 user ticks
Console: 2954 characters output
Keyboard: 0 keys pressed
Powering off...
cse20189999@cspro10:~/pintos/src/threads$
```

서강대학교
SOGANG UNIVERSITY

Running Pintos

- If you face the error like below, check the current directory where you run pintos.

- Since the current directory is src/utils, Pintos cannot find its kernel and error occurs.

- If you execute Pintos in src/threads, Pintos will find the kernel in src/threads/build/kernel.bin

Project Test

- Each project has its own test program
  - Test program is in src/tests.
  - You can use this program to test your implementation by yourself.
  - For example, in project 3, you can test by running 'make check' in src/threads/
    - ~/pintos/src/threads $ make check
    - PASS/FAIL will be printed for each test case

```
pass tests/threads/alarm-single
pass tests/threads/alarm-multiple
pass tests/threads/alarm-simultaneous
FAIL tests/threads/alarm-priority
pass tests/threads/alarm-zero
pass tests/threads/alarm-negative
FAIL tests/threads/priority-change
FAIL tests/threads/priority-donate-one
FAIL tests/threads/priority-donate-multiple
FAIL tests/threads/priority-donate-multiple2
```

※ src/threads/build/results

서강대학교
SOGANG UNIVERSITY

Requirements

Project #0

1. In **CSPRO9 or CSPRO10 (not CSPRO)** server, run <mark>**$pintos -v -- -q run alarm-multiple**</mark> and **capture the result of it** (you can just capture the last few lines of the result, **but your ID should be shown in the capture**).

2. **If your ID is not shown in the capture, deduct 10 points.**
    If "Powering off..." is not shown in the capture, deduct 70 points.
    If the Pintos doesn't quit properly (Kernel panic or other errors), deduct 70 points.

3. Use your own account in the server. (Don't borrow other's account.)

4. **Due Date: 09/20 23:59**
    Late submission is allowed up to 3 days (~09/23) and **10% of point will be deducted per day**

5. **Submit the capture file on e-class website**
    (Please use .jpg or .png extensions. Do not use other formats.)

6. **File name should be the following form:**
    os_prj0_ID#.jpg or os_prj0_ID#.png
    e.g.) os_prj0_20211234.jpg or os_prj0_20215678.png

서강대학교
SOGANG UNIVERSITY

Project #0

7. No hardcopy.

Reference Homepages

pintos	http://www.stanford.edu/class/cs140/projects/index.html
pintos document	http://www.stanford.edu/class/cs140/projects/pintos/pintos.pdf