

# **OS Project 3**

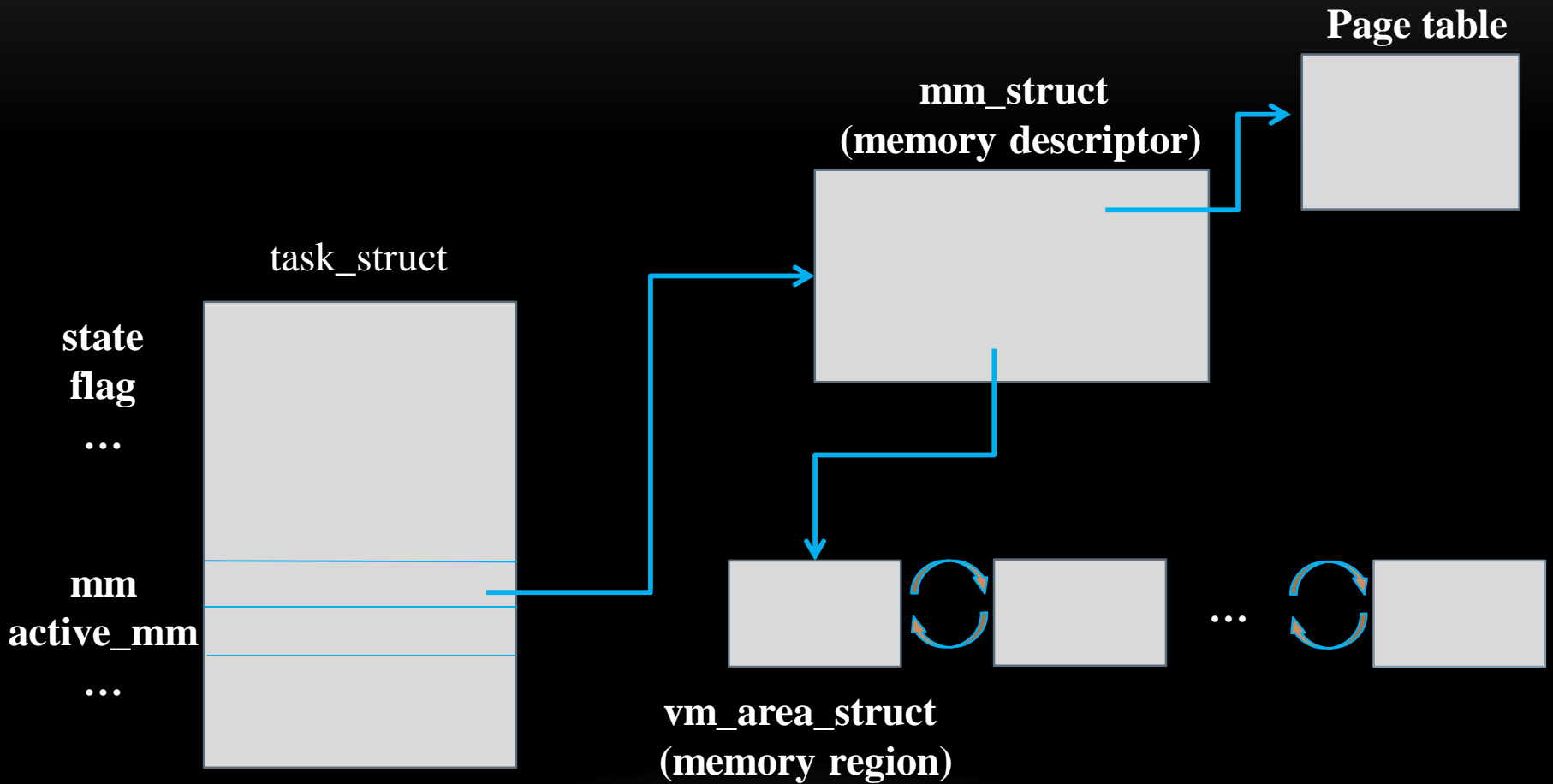
## **Hints**

---

**Advisor: Tei-Wei Kuo**

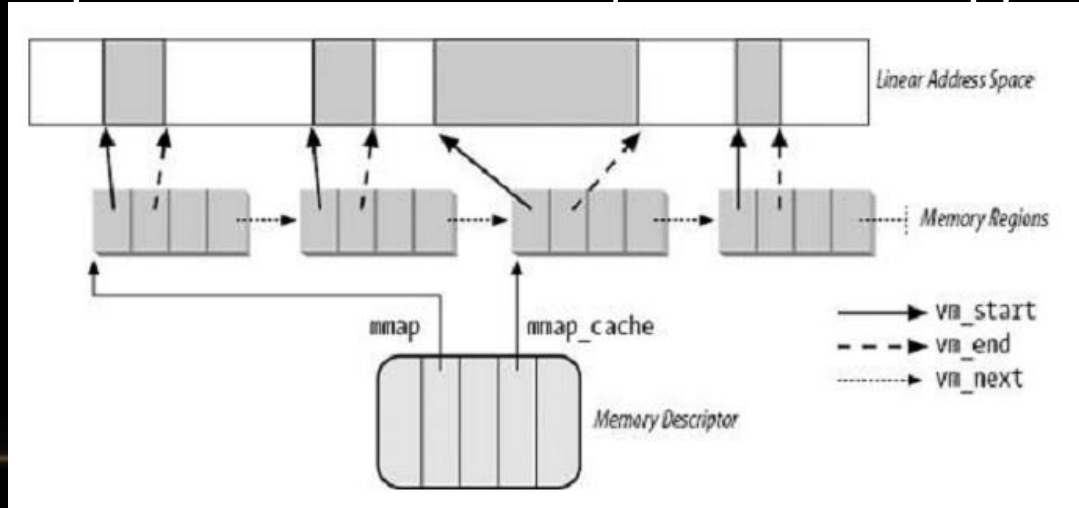
**Speaker: Che-Wei Tsao**

# Process Address Space



# The Data Structure of Memory Region

- `vm_start` – first linear address inside the region
- `vm_end` – first linear address after the region
- `vm_flags` – the access rights of the region
- `vm_ops` (`vm_operations_struct`) – pointer to the methods of the region
- `vm_file` – pointer to the file object of the mapped file, if any



# Memory Region Operations

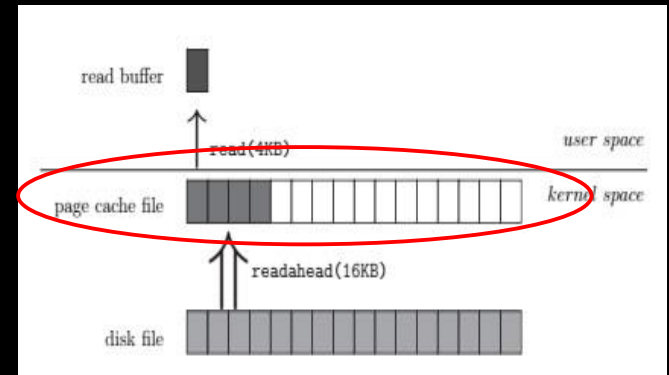
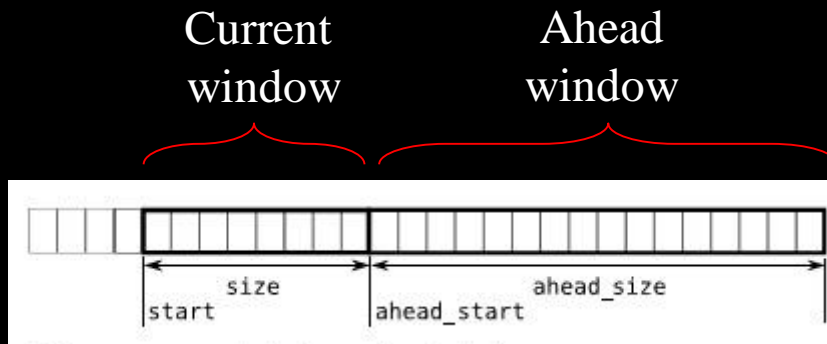
- `vm_operations_struct` [//include/linux/mm.h](#)
  - `void (*open)(struct vm_area_struct* area)`
  - `void (*close)(struct vm_area_struct* area)`
  - `int (*fault)(struct vm_area_struct* area, struct vm_fault* vmf);`
- File-backed memory regions will use a generic memory region operation
  - `generic_file_vm_ops` [//mm/filemap.c](#)
    - `.fault = filemap_fault`

```
struct vm_fault {  
    .....  
    unsigned int flags;  
    .....  
    pgoff_t pgoff;  
    void ____user *virtual_address;  
    .....  
    struct page *page;  
    .....  
};
```

# Readahead Scheme (kernel 2.6)

- On sequential access
  - Ahead window size =  $2 * \text{Current window}$   
(the size growing is stopped when reaching max size)
  - Whenever a request is crossing ahead window, it becomes current window and new request I/O is triggered to make new ahead window
- On random access
  - Ignore the ahead window
  - The size of the next ahead window is shrinked

- Sequential:  
file pointer = 0 or  
the page is the next page  
of last page fault



- Cache hit – minor page fault
- Cache miss – major page fault

# Another Options for Bonus

- Design your own readahead algorithm
  - Fine-tuned for the given test program
- Based on the original readahead algorithm
  - You can give kernel some advice about the access pattern
    - `madvise` function - give advice about use of memory
  - Code instrumentation in the test program is only allowed with this function

# Test Program

- Input.log
  - A random generated file
  - 128 MB
- test.c test.h
  - Map input.log into process address space
  - Read the first integer of a page specified by an index array
- Syslog.sh
  - Write message to system log

# Scoring of Project 3

- Revise mm/filemap.c/filemap\_fault() for pure demand paging (60%)
- Report (40%)
  - At most 4 pages
  - Trace code
  - Compare pure demand paging with the readahead algorithm by a case study
    - A test program is given



# Scoring of Project 3 (cont)

- Bonus (at most 20%)
  - Implement your own readahead algorithm (15%)
    - Either one of the following conditions must be satisfied
      - # of page faults  $<$  # of default page faults
      - # of RSS (Resident Set Size)  $<$  # of default RSS
  - Report (5%)
    - Additional 2 pages at most


# Submission Rules

- Project deadline: **2015/06/17 23:59**
  - Delayed submissions yield severe point deduction
- Upload your team project to the FTP site.
  - **FTP server: 140.112.28.132 (SFTP)**
  - 請用 **anonymous** (匿名) 方式登入
- The team project should
  - Contain the whole “**linux3.2.54/**” directory
  - Contain your page fault syslog
  - Contain your test program syslog
  - Contain your report (PDF, within 6 pages)
  - Be packed as one file named “OSPJ3\_Group##.tar.xz”
- **DO NOT COPY THE HOMEWORK**

# References

- Understanding the Linux Virtual Memory Manager
- Understanding the Linux kernel, 3rd
- LinuxMM,  
<http://linux-mm.org/>
- Linux Cross Reference,  
<http://lxr.free-electrons.com/>
- Kernel Parameters,  
<http://lxr.free-electrons.com/source/Documentation/kernel-parameters.txt>
- Debugging by printing,  
[http://elinux.org/Debugging\\_by\\_printing](http://elinux.org/Debugging_by_printing)

# Contact TAs

- If you have any problem about the projects, you can contact TAs by the following ways:
- Facebook: NTU OS2015 Spring Group 
  - <https://www.facebook.com/groups/920624997989865/>
- E-mail:
  - Che-Wei Tsao: d02944011@csie.ntu.edu.tw