SWE3004: Operating Systems
prof. Euiseong Seo

Project 2. Thread

2019.4.3 (Wed.)

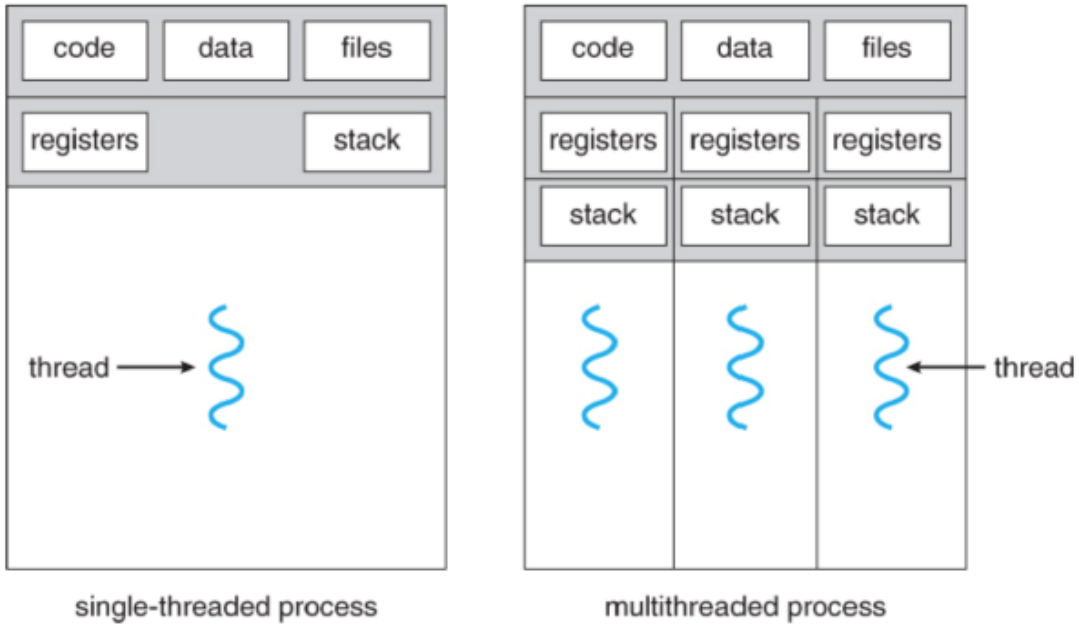
TAs
김종석(ks77sj@gmail.com) /
최동규(gmj03003@gmail.com)

Project Plan

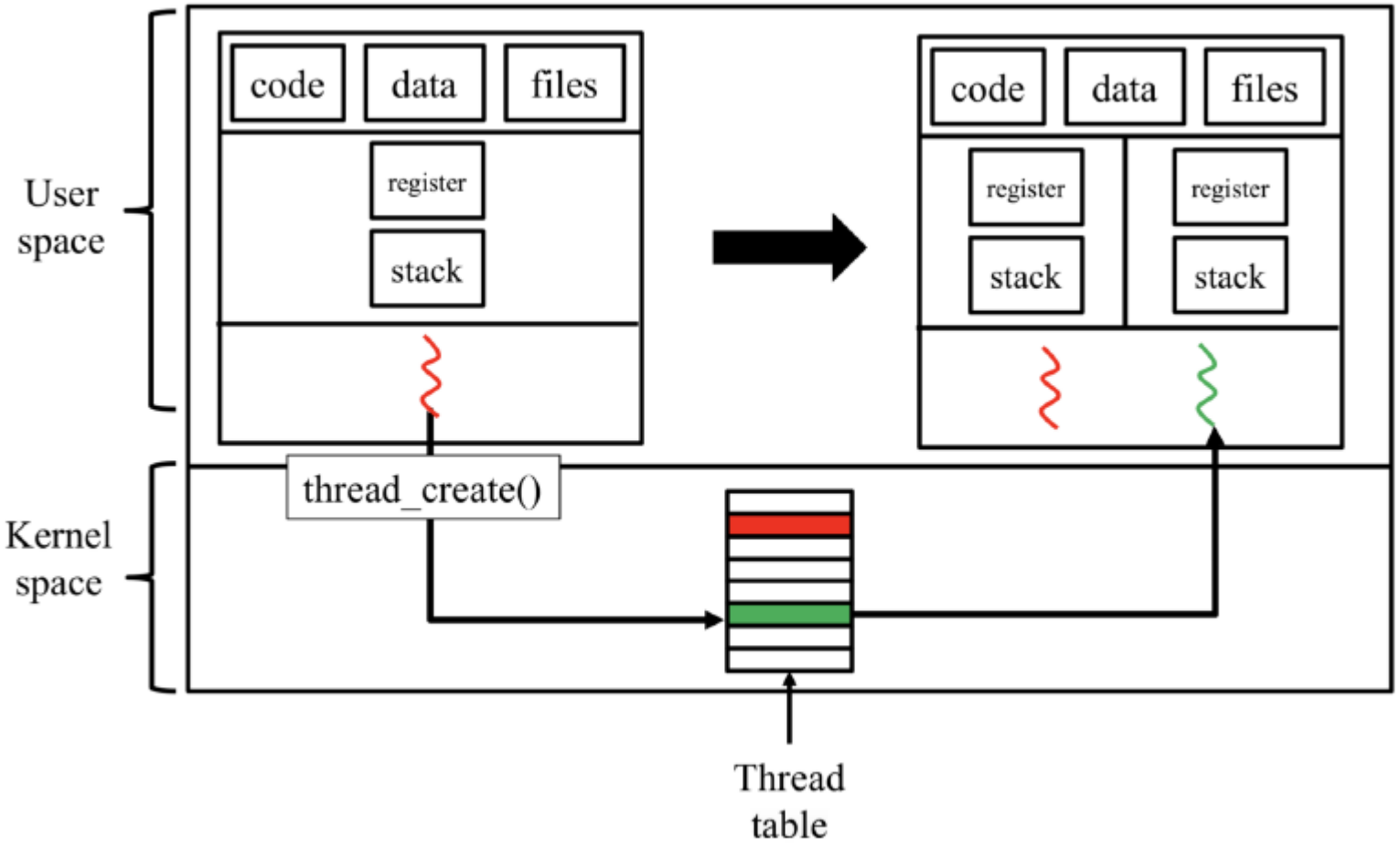
- Total 7 projects
  0) Starting xv6 operating system (5%)
  1) System call (10%)
  2) Thread (15%)
  3) Syncronization (15%)
  4) Scheduling 1(10%)
  5) Scheduling 2(15%)
  6) Page fault handler (15%)
  7) Copy on Write (15%)

Supporting Threads on Xv6

- The original xv6 process is single-thread
- Multi-thread envirionment
  - Each thread has its own stack
  - Every threads shares code, data and other resources such as open files



Supporting Threads on Xv6



Project 2. Thread

- Make 3 API with thread supporting
  - int thread_create(void *(*function)(void *), void *arg, void *stack)
  - void thread_exit(void *retval)
  - int thread_join(int tid, void **retval)

- Implement method for getting thread ID
  - int gettid(void)

thread_create()

- Synopsis
  - Create a new thread at calling process
  - int thread_create(void *(*function)(void *), void *arg, void *stack)

- Return value
  - Return the thread ID(tid) of the new thread
  - If err, return -1

thread_create()

- The new thread starts execution in invoking function.
- Arg is passed as the argument of function.
- Stack is the pointer to call stack of new thread.
- All threads in a process have same pid & priority.
- Initial thread in a process is a main thread which has tid ‘1’.
- A process can have maximum 8 threads.

thread_exit ()

- Synopsis
  - Terminate the calling thread
  - void thread_exit(void *retval)

thread_exit ()

- Each thread save retval at thread_exit().
- Thread state transfers to ZOMBIE.
- Thread resources are retrieved at thread_join.
- Exiting thread may wake up threads which have a same pid.

thread_join ()

- Synopsis
  - Join with terminated thread
  - thread_join(int tid, void **retval)

- Return value
  - If success, return 0
  - If there’s no thread with input tid, return -1

thread_join ()

- Wait thread specified with tid to terminate.
  - Caller may sleep until corresponding thread terminated.
  - If thread has already terminated, return immediately.
- Copy the exit status of the target thread into the location pointed to by retval.
- The call stack of the terminated thread should be freed by the calling thread.

gettid()

- Returns caller’s thread ID.
- In multi-thread process, all threads have the same PID.
- Each thread has a unique TID within a process.

Things to Consider (1)

- We assume that each thread always terminates by calling thread_exit().
- If the main thread terminates or any thread calls exit(), whole process is terminated. In this case, all threads should be terminated as well. Also, address space should be freed and open files should be closed.

Things to Consider (2)

- When a thread calls thread_exit(), the thread remains in ZOMBIE state until another thread calls thread_join().
- Any thread within a process can invoke thread_join() for another thread.
- All threads within a process should return the same process ID. Thread IDs are guaranteed to be unique only within a process.

Hint

- You can download a test file on Project site.
  - Or using wget
    - $wget http://csl.skku.edu/uploads/SWE3004S19/project2_test.zip
  - Decompress and use as a user program to check if printing OK is available.
- Refer to fork(), wait(), and exit() in proc.c file.
- In order to implement the thread_create(), you may need to know about procedure, which you learn from the system program.

Submission

- You need to submit a document.
- Just write how you implemented your code.
- You can use English or Korean.

Submission

- Send your code file (xv6-project-2-studentID.tar.gz) and document file to ks77sj@gmail.com
- Please send a mail with tittle including [SWE3004-P2]
  - Ex) [SWE3004-P2] 2014111111-project2
- PLEASE DO NOT COPY
  - YOU WILL GET F GRADE IF YOU COPIED
- Due date: 4/10(Wed.), 23:59:59 PM
  - Delays are allowed only one week from the deadline. And there will be up to -40% penalty.

Questions

- If you have questions, please email to TA
- You can also visit #85533. Please email TA before visiting