EEE3052: Introduction to Operating Systems

Fall 2017

Project #3

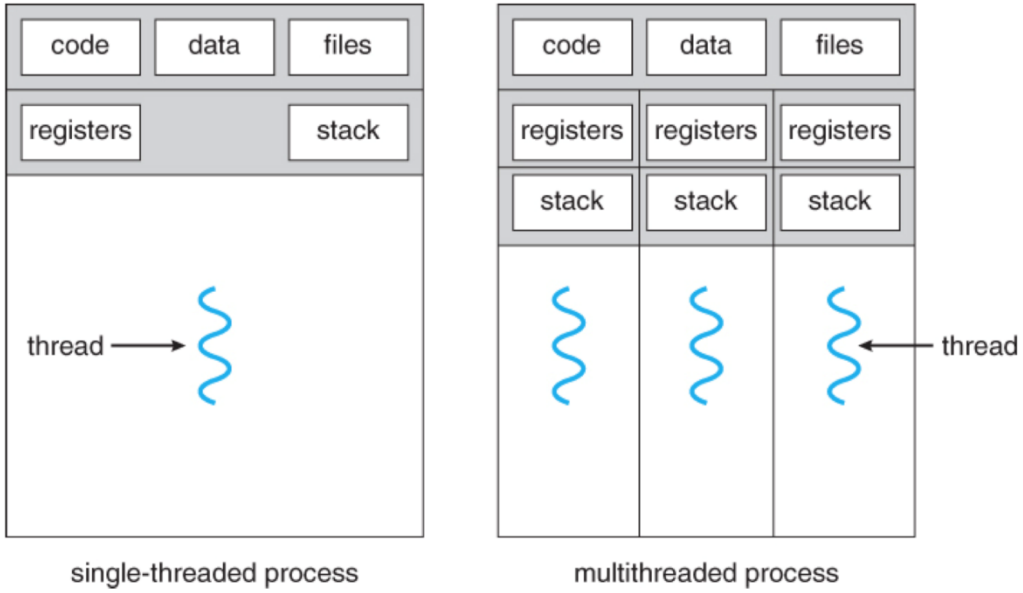
Project Plan

- 4 projects
  0) Install Xv6
  1) Process management
  2) Virtual memory
  3) Synchronization
     - Thread (11/13 ~ 11/19)
     - Mutex & Condition variable (11/20 ~ 12/3)
  4) File system

Supporting Threads on Xv6

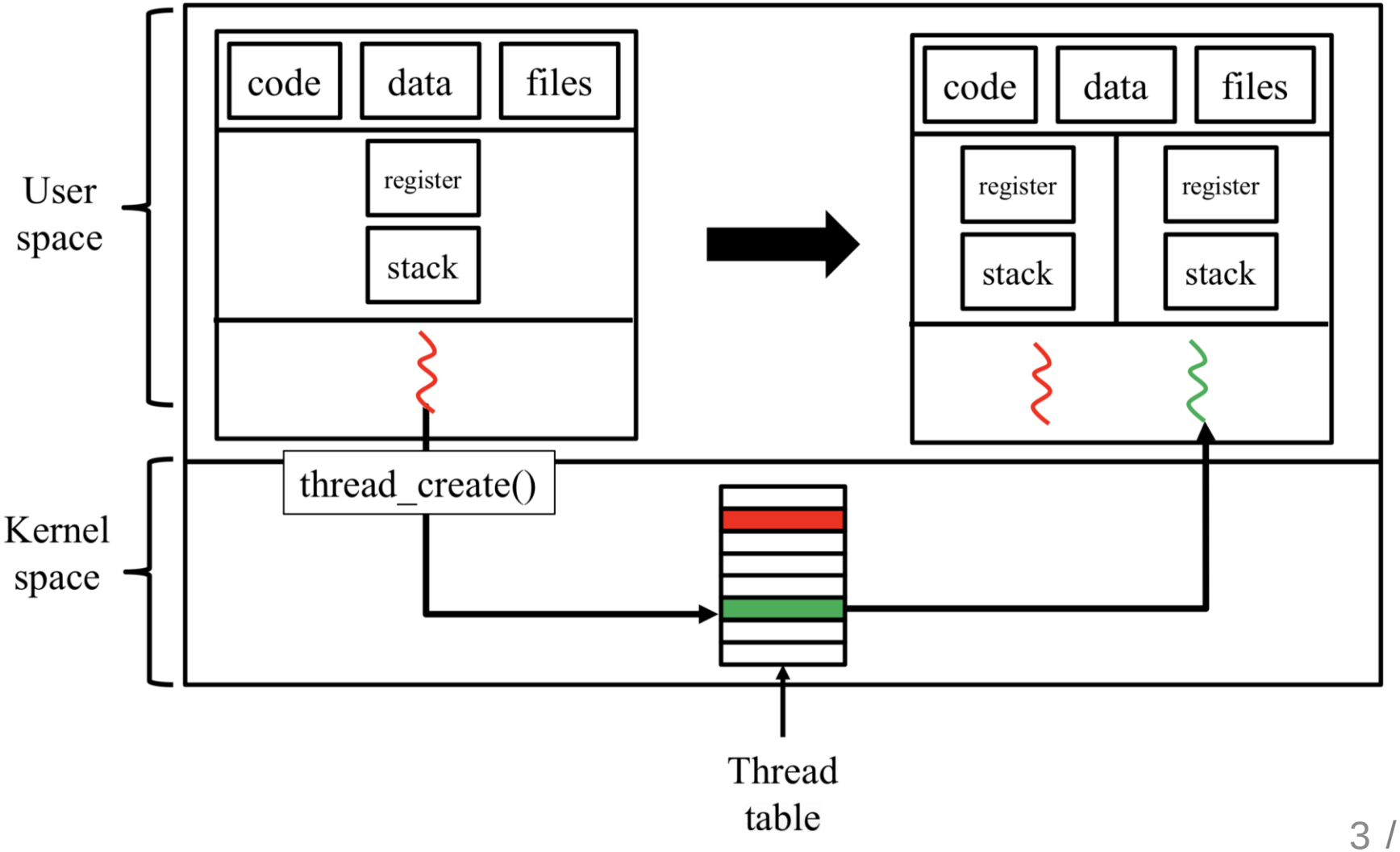
- The original Xv6 process is single-thread

- A multi-threaded process consists of multiple threads
  - Each thread has its own call stack
  - Every thread shares code, date, and other resources such as open files



https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/4_Threads.html

Supporting Threads on Xv6



Project #3-1 – Thread

- Make 3 thread API
  - 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()

- Name
  - thread_create: create a new thread at calling process
- Synopsis
  - int thread_create(void *(*function)(void *), void *arg, void *stack);
- Return value
  - Return the thread ID (tid) of the new thread.
  - At one process, tid is guaranteed to be unique
  - On err, return -1

thread_create() (Cont.)

- The new thread starts execution by invoking *function*.
- *arg* is passed as the sole 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 tid = 1 & parent of others.
- A process can have maximum 8 threads.

thread_exit()

- Name
  - thread_exit: terminate calling thread
- Synopsis
  - void thread_exit(void *retval);
- Return value
  - This function does not return to the caller

thread_exit() (Cont.)

- Each thread save *retval* at *thread_exit()*.
- Thread state transfers to ZOMBIE.
- Thread resources are retrieved at *thread_join()*.
- Exiting thread may wake up parent thread.

thread_join()

- Name
  - thread_join: join with a terminated thread
- Synopsis
  - int thread_join(int tid, void **retval);
- Return value
  - On success, return 0.
  - If there’s no thread with input tid, return -1.

thread_join() (Cont.)

- Wait for the thread specified by *tid* to terminate.
  - Caller may sleep until 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.
  - Resources should be retrieved at this point.

gettid()

- The gettid() function returns caller’s thread ID.
- In a multi-thread process, all threads have the same PID.
- However, each thread has a unique TID within a process.

Makefile

- CPU number is fixed to 2.

```
ifndef CPUS
CPUS := 2
endif
QEMUOPTS = -drive file=fs.i
```

- Add 6 test files.

Round Robin Priority Scheduler

- Restore *yield()* in trap.c

```
if(myproc() && myproc()->state == RUNNING &&
   tf->trapno == T_IRQ0+IRQ_TIMER)
  yield();
```

File Descriptor

- Thread may not print your message.
- Before (proc.h)

```
struct file *ofile[NOFILE];  // Open files
```

- After (proc.h)

```
struct file **ofile;  // Open files
struct file *_ofile[NOFILE];  // Open files
```

- Add
  - *allocproc()*

```
p->ofile = p->_ofile;
```

  - *thread_creatr()*

```
p->ofile = p->parent->ofile;
```

Test Case

- 6 test cases.
- Test cases may not have an effect on your grade.
- Just use for guide & test.

Things to Consider (1)

- A process can call *thread_exit().*
- A process can call AND success thread_create() more than 8 times.
- *setnice()*
  - We fix rule that all thread in process have same priority.
- Thread can call *thread_create()*.
  - *thread_create()* can be nested.

Things to Consider (2)

- Thread can call *thread_join()*.
  - *thread_join()* can be nested.

- Thread can call *exit()*.
  - At this time, process & all threads are terminated.

Reference

- Our thread interface is a simplified version of POSIX Pthreads interface.

- Refer to manual pages on
  - *pthread_create()*
  - *pthread_exit()*
  - *pthread_join()*

Submit

- Send email to jaehyun.song@csl.skku.edu
  - Title: [EEE3052]Project-3_1-studentID-name
  - File name: studentID-3_1.tar.gz

- Due date
  - 2017/11/19 Sunday 23:59:59
  - Penalty 10% of each project score per one day

- TA contact
  - jaehyun.song@csl.skku.edu
  - minwoo.ahn@csl.skku.edu