SSE3044 Introduction to Operating Systems

Prof. Jinkyu Jeong

# Project 1. System call

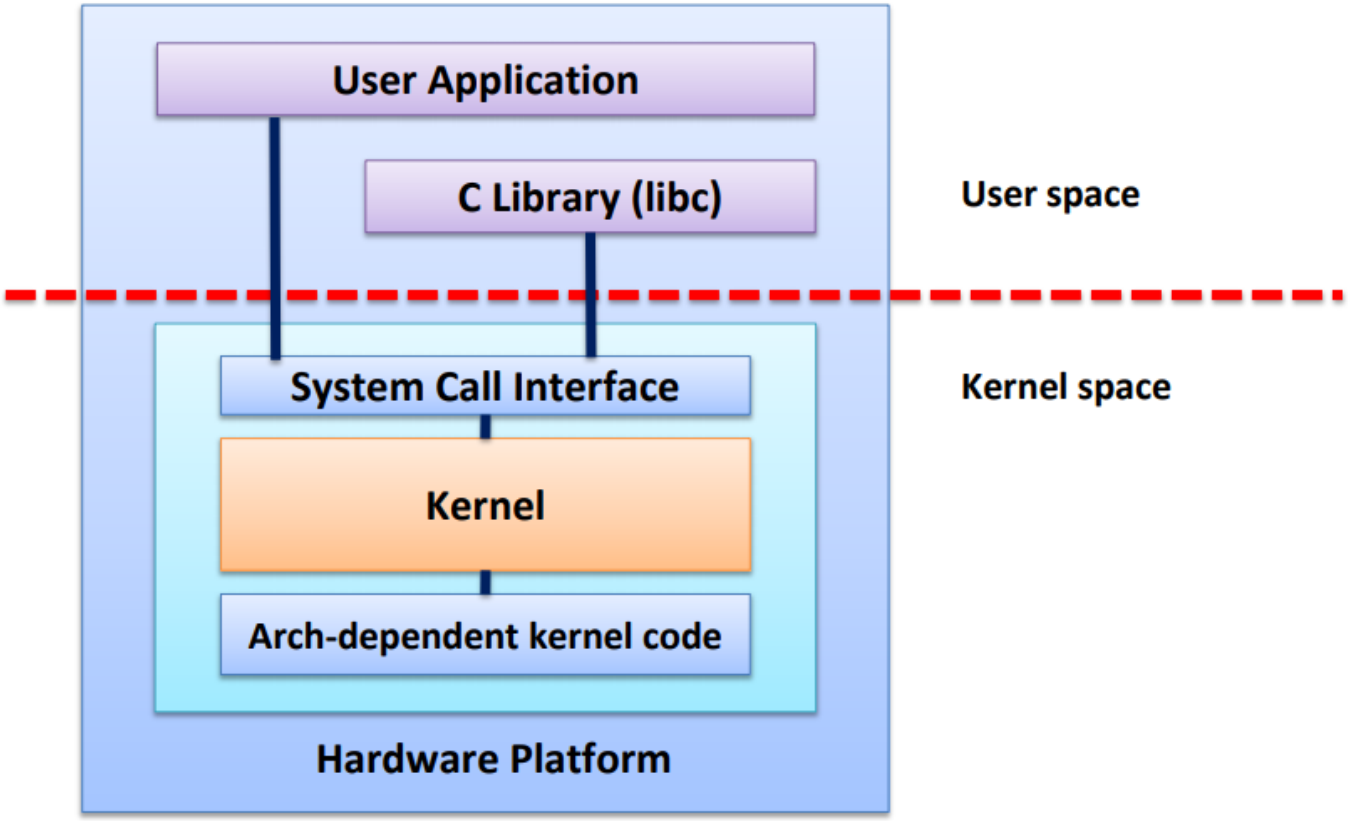
TA)
송원석(wonsuk.song@csi.skku.edu)
진승우(seungwoo.jin@csi.skku.edu)

# Project plan

- Total 6 projects
  0) Booting xv6 operating system
  1) System call
     - system call to set process priority
     - system call & user command (ps)
  2) CPU scheduling
  3) Virtual memory
  4) Page replacement
  5) File systems

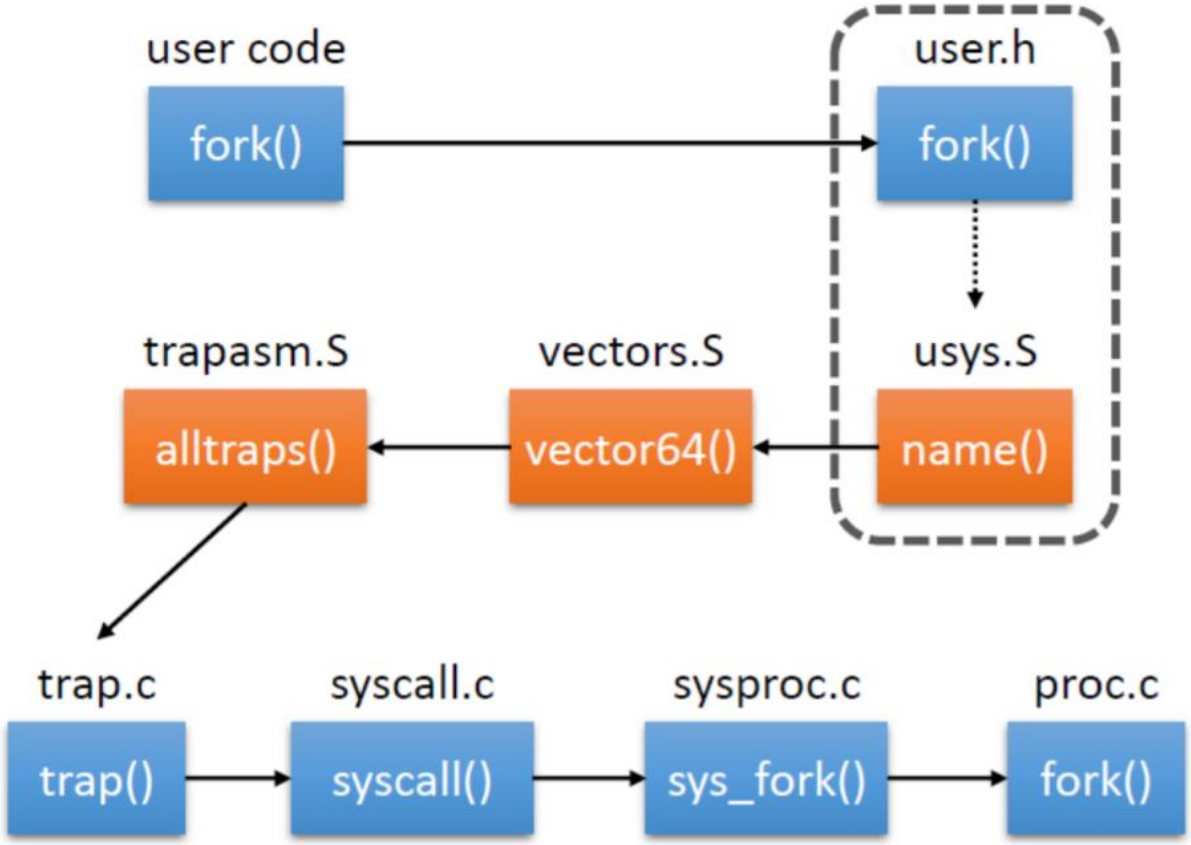
# What is system call?

- An interface for accessing the kernel from user space



# System call flow in xv6

- Ex) fork() system call



# Process priority

- Every process has a priority
- CPU Scheduling determines which process to run based on the priority of the process
- We will adjust these process priorities using the 'nice' value in xv6
- But, now in xv6 => not implemented!

# Process priority

- Make setnice() and getnice() system call
- User process set priority by using nice value (0 ~ 10)
  - setnice()
    - Set process priority
    - Argument: process ID, nice value
    - Return value
      - If set nice failed, return -1
      - Else, return 0
  - getnice()
    - Return process priority
    - Argument: process ID
    - Return value
      - If process corresponding ID does not exist, return -1
      - Else, return process's nice value

# Process priority - details

- Initial process's nice value must be 5
- Set nice to current process
- Child processes created through the fork() inherit the nice value from the parent process

- Wrong case => should return -1
  - Set nice to non-existing process (wrong pid)
  - Set wrong nice( < 0 or > 10) on current process
  - Get nice of non-existing process (wrong pid)

# Yield system call

- Cooperative Multi-tasking
  - Trust process to relinquish CPU to OS through traps
  - Provide special yield() system call

- Make yield system call
  - Yield function is already defined in xv6
    - In xv6, It periodically generates interrupts and gives CPU to other processes through yield function
  - Using already defined yield function, make yield system call

# System call & user command (ps) (1)

- Make system call to monitor process status (ps)
  - Print all process status
  - Contents
    - pid
    - State : RUNNING, RUNNABLE, SLEEPING
    - Priority(nice value)
    - Runtime
    - Current tick of system
  - And make user program ps.c using ps system call, modify the Makefile!

# System call & user command (ps) (2)

- Modify the Makefile, then you will see that ps program in the result of 'ls' command

```
UPROGS=\
        _cat\
        _echo\
        _forktest\
        _grep\
        _init\
        _kill\
        _ln\
        _ls\
        _mkdir\
        _rm\
        _sh\
        _stressfs\
        _usertests\
        _wc\
        _zombie\
        _ps\
        _test1\
        _test2\
```

```
$ ls
.               1 1 512
..              1 1 512
README          2 2 2327
cat             2 3 13476
echo            2 4 12552
forktest        2 5 8260
grep            2 6 15304
init            2 7 13140
kill            2 8 12592
ln              2 9 12496
ls              2 10 14712
mkdir           2 11 12616
rm              2 12 12592
sh              2 13 23228
stressfs        2 14 13268
usertests       2 15 56148
wc              2 16 14128
zombie          2 17 12320
ps              2 18 12276
test1           2 19 14140
test2           2 20 13528
```

```
$ ps
name    pid     state           priority        runtime         tick 149
init    1       SLEEPING        5               2
sh      2       SLEEPING        5               0
ps      3       RUNNING         5               0
```

# Modify project template code

- To test your implementation, modify some code
- Modifications
  - To set xv6's cpu number 2 to 1
    - Makefile -> CPUS := 1

```
ifndef CPUS
CPUS := 1
endif
```

  - Remove according lines of trap() in trap.c
    - => disable automatic yield

```
// Force process to give up CPU on clock tick.
// If interrupts were on while locks held, would need to check nlock.
/*if(myproc() && myproc()->state == RUNNING &&
   tf->trapno == T_IRQ0+IRQ_TIMER)
  yield();
*/
```

# Submission

- Please implement 4 system calls and user command(ps)
- Compress your source code and upload on i-Campus
- How to compress your project
  - If you insert the user program, Modify the 'EXTRA' in Makefile
  - make dist
  - make tar
  - then, tar.gz file will be generated automatically
  - Rename to studentID-project1.tar.gz and upload on i-Campus

# Submission

- File format
  - StudentID-project1.tar.gz

- PLEASE DO NOT COPY
  - YOU WILL GET F GRADE IF YOU COPIED

- Due date: 4/7(Tue.), 23:59:59 PM
  - -25% per day for delayed submission

# Questions

- If you have questions, please ask in Piazza
- You can also visit Semiconductor Building #400509
  - Please e-mail TA before visiting
- Reading xv6 commentary will help you a lot
  - http://csl.skku.edu/uploads/SSE3044S20/book-rev11.pdf