Operating System

Project #4
16. 11. 14
Project Plan

• 5 projects
  - Install Xv6
  - System call + scheduling
  - Virtual memory (stack growth + COW)
  - Thread-support
  - Concurrency

• Single-handed project
Thread-Support on Xv6

• Xv6 process is single-threaded

• Multithreaded process consists of one or more threads
  – Each thread has its own call stack
  – Every thread shares code, data, and other resources such as open files
Thread-Support on Xv6
Thread-Support on Xv6 – thread_create()

- Name
  thread_create - create a new thread

- Synopsis
  ```c
  int thread_create(void *(*function)(void *), int priority, void *arg, void *stack);
  ```

- Description
  The thread_create() starts a new thread in the calling process. The new thread starts execution by invoking function(); arg is passed as the sole argument of function(). priority is the scheduling priority of the new thread (0~40). stack is the pointer to call stack of new thread.

- Return value
  Return the thread ID(tid) of new thread. tid is guaranteed to be uniqued within a process. On error, return -1.
Thread-Support on Xv6 – thread_exit()

- **Name**
  thread_exit – terminate calling thread

- **Synopsis**
  void thread_exit(void *retval);

- **Description**
  The thread_exit() terminates calling thread and returns a value via retval that is available to another thread in the same process that calls thread_join().

- **Return value**
  This function does not return to caller.
Thread-Support on Xv6 – thread_join()

• Name
  thread_join – join with a terminated thread

• Synopsis
  int thread_join(int tid, void **retval);

• Description
  The thread_join() waits for the thread specified by tid to terminate. If that thread has already terminated, then thread_join() returns immediately. thread_join() copies exit status of the target thread into the location pointed by *retval. The call stack of terminated thread should be freed by the calling thread.

• Return value
  On success, return 0. If there is no thread with input tid, return -1.
Thread-Support on Xv6 – gettid()

- Name
  gettid – get thread ID

- Synopsis
  int gettid(void);

- Description
  The gettid() returns thread ID of caller. If the process is a single-threaded process, thread ID is same as the process ID. In a multi-threaded process, all threads have same process ID, but each one has a unique thread ID within a process.

- Return value
  Return the thread ID of calling thread.
Thread-Support on Xv6 – getpid()

- **Name**
  getpid – get process identification

- **Synopsis**
  ```c
  int getpid(void);
  ```

- **Description**
  The getpid() returns process ID of caller. On multi-threaded process, every thread of the same process returns same process ID.

- **Return value**
  Return the process ID of calling process.
Thread-Support on Xv6

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

- Open files are shared among threads. If thread A opens a file, the file can be also accessed by another thread B (in the same process) using same file descriptor. Files opened by thread A need not be closed automatically when thread A terminates.
Thread-Support on Xv6

- When a thread calls thread_exit(), the thread remains in zombie state until another thread calls thread_join().

- There is no parent-child relationship among thread. Any thread 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.

- Maximum number of threads per process is limited to 8 (including main thread). (param.h → NTHREAD)
Project #4 – Thread-Support

- Implement following system calls in xv6
  - thread_create()
  - thread_exit()
  - thread_join()
  - gettid()

- Modify following system call to support threads
  - getpid()

- Implement priority scheduler that supports threads
Project #4 – Thread-Support

• Implement thread-support in xv6

• Submit a tar.gz file

• Send email to T.A
  – [SSE3044]Project#4-YOURID-YOURNAME
  • ex) [SSE3044]Project#4-2016710580-leegyusun
  – Email address: lgs0409@naver.com

• Due date
  – 2016-11-27(Sun) PM 23:59
  – -10% per day (until 11/30)