Recitation 10
Process vs. Threads

**PROCESS**

- User Address Space:
  - Stack
    - `routinel var1() var2()`
  - Text
    - `main() routinel() routine2()`
  - Data
    - `arrayA arrayB`
  - Heap

**THREADS**

- User Address Space:
  - Stack
    - `routine2() var1 var2 var3`
    - `routine1() var1 var2`
    - `main() routinel() routine2() ...`
  - Text
    - `main()`
  - Data
    - `arrayA arrayB`
  - Heap
Fork() v/s pthread_create()
Thread-Specific Resources

- Each thread has its own:
  - Thread ID (integer)
  - Stack, registers, program counter
  - Errno
  - arithmetic/buffer overflow

- Threads within the same process can communicate using shared memory - *Must be done carefully!*
pthread_create

- Creating a thread is like a combination of fork() and exec()
- `#include <pthread.h>`
  ```c
  int pthread_create(
      pthread_t *thread,
      pthread_attr_t *attr,
      void *(*function)(void *),
      void *arg);
  ```
- thread is the thread ID, attr an attribute set, function is the function to be called with arg
- Compile/Link with `-D_REENTRANT -lpthread`
pthread_join(), pthread_exit()

- `int pthread_join(pthread_t thread, void **retval);`
  - wait until a thread exits
- `void pthread_exit(void *retval);`
  - exit from current thread
pthread_join(), pthread_exit()

- int pthread_join(pthread_t thread, void **retval);
  - wait until a thread exits
- void pthread_exit(void *retval);
  - exit from current thread
Detaching threads

- #include <pthread.h>
  
  int pthread_detach (pthread_t thread);

- A detached thread cannot be joined – it will just go away when it exits

- You cannot detach a thread that some other thread is joining it
pthread_attr_t *attr

- Thread attributes can be set using attr, including detached state and scheduling policy.
- You can specify NULL and get the system defaults
- Also, you can override to specify custom options.
- `int pthread_attr_setscope(pthread_attr_t *attr, int scope);`
- `int pthread_attr_getscope(const pthread_attr_t *attr, int *scope);`
- `int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy);`
- `int pthread_attr_getschedpolicy(const pthread_attr_t *attr, int *policy);`