HW09

2019203102 유지성

Thread Create

```
8       printf("Hello. I'm main thread: %ld\n", pthread_self());
9       if (pthread_create(&tid, NULL, myThread, (void *)&count)) {
10          perror("pthread_create() error!");
11          return -1;
12      }
```

main thread의 TID를 출력 후 새로운 thread 생성, count를 thread의 인자로 설정

```
13      pthread_join(tid, (void **)&status);
14      printf("Thread %ld exit\n", tid);
```

생성된 thread가 종료되기를 기다림.

```
17  void *myThread(void *arg) {
18      int i = 0;
19      int count = *(int *)arg;
20      int status = 0;
21      printf("Hello. I\'m new thread: %ld\n", pthread_self());
22      for (i = count; i > 0; --i) {
23          printf("%d ...\n", i);
24      }
25      pthread_exit((void *)&status);
26  }
```

TID를 출력하고 for문으로 문자를 출력하는 thread 함수. 작업이 끝나면 pthread_exit로 thread 종료.

Thread Cleanup

```
18  void *myThread(void *arg) {
19      int i = 0;
20      int count = *(int *)arg;
21      int status = 0;
22      printf("Hello. I\'m new thread: %ld\n", pthread_self());
23      pthread_cleanup_push(cleanupHandler, "Handler 1");
24      pthread_cleanup_push(cleanupHandler, "Handler 2");
25      for (i = count; i > 0; --i) {
26          printf("%d ...\n", i);
27      }
28      pthread_cleanup_pop(1);
29      pthread_cleanup_pop(1);
30      pthread_exit((void *)&status);
31  }
```

handler 1, handler 2가 stack에 저장되어서 pop할 때 handler 2, handler 1 순으로 실행된다.

```
32  void cleanupHandler(void *arg) { printf("Cleanup: %s\n", (char
        *)arg); }
```

cleanupHandler가 호출될 때 문장을 출력한다.

Mutex

```
12      pthread_mutex_init(&mutex, NULL);
13      if (pthread_create(&tid1, NULL, myThread1, NULL)) {
14          perror("pthread_create() error!");
15          goto END;
16      }
17      if (pthread_create(&tid2, NULL, myThread2, NULL)) {
18          perror("pthread_create() error!");
19          goto END;
20      }
21      pthread_join(tid1, (void **)&status);
22      pthread_join(tid2, (void **)&status);
```

mutex를 초기화하고, thread를 두 개 생성

```
21      pthread_join(tid1, (void **)&status);
22      pthread_join(tid2, (void **)&status);
23  END:
24      pthread_mutex_destroy(&mutex);
25      return 0;
```

thread 두 개가 종료될 때까지 기다린 후 mutex를 destroy

```
27  void *myThread1(void *arg) {
28      int i = 0;
29      int status = 0;
30      pthread_cleanup_push(cleanupHandler, "Thread 1");
31      pthread_mutex_lock(&mutex);
32      puts("Thread 1");
33      for (i = 0; i < 5; ++i) {
34          sharedNum += i;
35          printf("%d ", sharedNum);
36      }
37      puts("");
38      pthread_cleanup_pop(1);
39      pthread_exit((void *)&status);
40  }
```

mutex를 lock하고 공유 자원을 이용해 숫자를 출력, 출력이 완료되면 cleanupHandler 호출

```
55  void cleanupHandler(void *arg) {
56      pthread_mutex_unlock(&mutex);
57      printf("Cleanup: %s\n", (char *)arg);
58  }
```

mutex를 unlock해 사용할 수 있게 함.