KWANGWOON UNIVERSITY

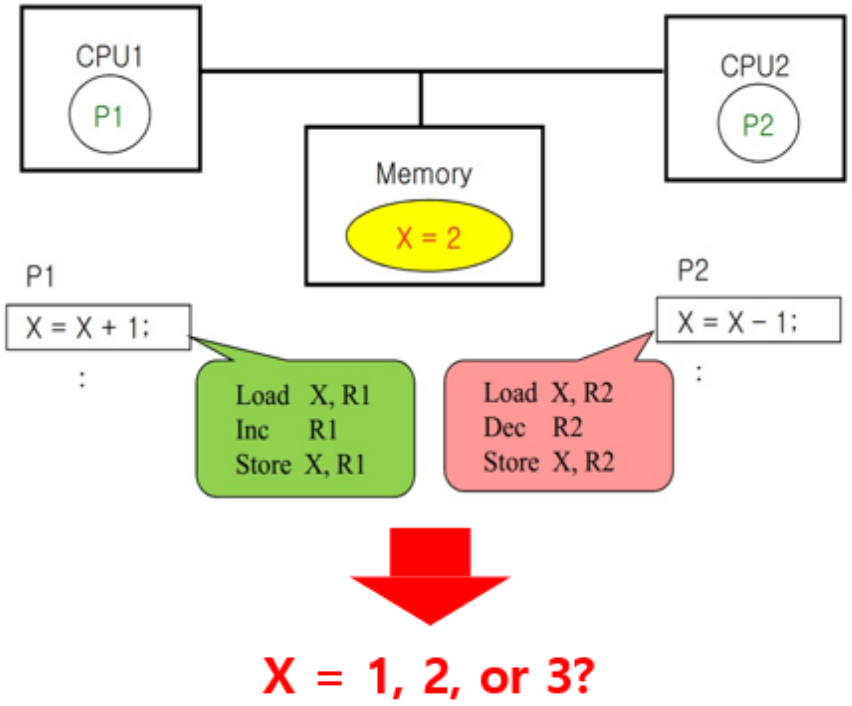
2022년 1학기 **시스템프로그래밍실습** 14주차

Mutex

System Software Laboratory
College of Software and Convergence
Kwangwoon Univ.

Synchronization (1/2)

- **동기화의 필요성**
  - 멀티프로세서 환경 또는 시분할 방식에서, 병렬성(parallelism)을 활용하여 처리율, 응답 시간 등의 성능 향상을 얻을 수 있음
  - 다만, 공유 자원에 동시에 접근 하여 경쟁 조건이 발생할 수 있으므로 이를 방지하여야 함
    - 공유 자원 (shared resource)
      - 시스템 자원의 대부분이 공유될 수 있음
    - 경쟁 조건 (race condition)
      - 하나 이상의 프로세스가 동일한 자원을
        - 접근하기를 원하는 상태



Synchronization (2/2)

- **동기화 순서**
  1. 임계 구역 (critical section)을 정의
     - 공유자원을 접근하기 위한 코드의 일부
  2. 상호 배제 (mutual exclusion) 메커니즘의 사용
     - 한 시점에 하나의 프로세스만 공유 자원을 접근할 수 있음.

- **대표적인 동기화 방법**
  - 락, 세마포어, 파이프 등

- **주의 사항**
  - 데드락이 일어나지 않도록 해야 함
    - 데드락(deadlock): 절대 발생하지 않을 일을 무한정 기다리는 현상

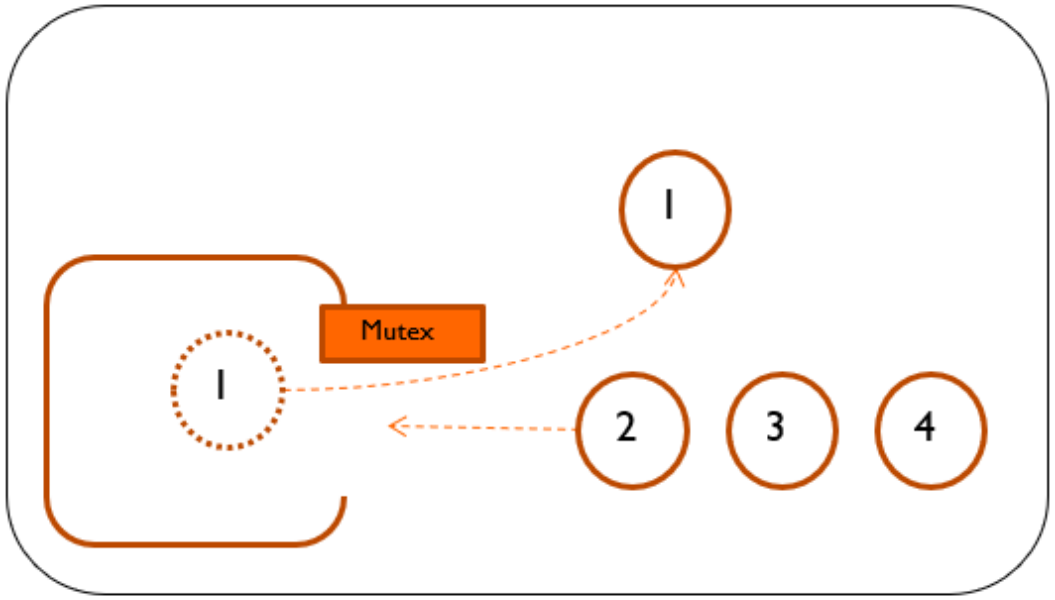
Spin Lock

- **스핀 락 (spin lock)**
  - 리눅스 커널에서 가장 일반적인 락킹 기법.
  - 다른 thread가 이미 가진 스핀 락을 얻으려고 시도한다면?
    - 그 락을 얻을 때까지 busy loop 로 대기

- **특징**
  - 문맥 교환이 필요하지 않음
  - Busy loop 로 인한 CPU 소모
    - 짧은 기간 대기에 효과적임
    - 스핀 락을 사용하는 시간이 두 번의 문맥교환 시간보다 짧은 것이 효율적
  - Linux에서는 SMP 환경에서만 사용
  - Symmetric Multi-Processing (SMP)
    - 두 개 이상의 프로세서가 한 개의 공유된 메모리를 사용하는 구조

Mutex: Mutual Exclusion (1/6)

- **Mutual Exclusion**
  - 여러 thread 간 동시간 접근을 허용하지 않도록 하는 것 (lock을 얻지 못하면 대기상태)



Mutex: Mutual Exclusion (2/6)

- **Mutex 제약조건**
  - Mutex의 사용 카운트 값은 항상 1
  - Mutex를 얻은 곳에서만 다시 해제할 수 있음
  - 같은 Mutex를 재귀적으로 여러 번 얻을 수 없음
  - Mutex를 가지고 있는 동안에는 프로세스의 종료가 불가능
  - 주어진 공식 API를 통해서만 관리할 수 있음

Mutex: Mutual Exclusion (3/6)

- **Semaphore와 비교**
  - 접근 thread 수
    - Mutex ➔ only 1개

    : 오직 1개만 공유 자원에 접근 가능
    - Semaphore ➔1개 이상

    : 지정한 수 만큼 공유 자원에 접근 가능

  - 다른 thread의 Lock 해제
    - Mutex

    : Lock을 획득한 thread만 Lock 해제 가능
    - Semaphore

    : 현재 수행하는 thread가 아닌 다른 thread도 Lock 해제 가능

Mutex: Mutual Exclusion (4/6)

- **Mutual Exclusion**
  - 여러 thread 간 공유 자원의 동시간 접근을 허용하지 않도록 하는 것

- **pthread의 Mutex 관련 함수**

```
int pthread_mutex_init (pthread_mutex_t *mutex,
                        const pthread_mutexattr_t *mutexattr)
int pthread_mutex_lock (pthread_mutex_t *mutex)
int pthread_mutex_unlock (pthread_mutex_t *mutex)
int pthread_mutex_destroy (pthread_mutex_t *mutex)
```

Mutex: Mutual Exclusion (5/6)

- **APIs**
  - pthread_mutex_init()
    - 사용할 mutex 변수를 초기화 함

```
#include <pthread.h>

int pthread_mutex_init(pthread_mutex_t * mutex, const pthread_mutex_attr  *attr);
```

      - pthread_mutex_t * mutex: 사용할 mutex 변수의 주소 값
      - const pthread_mutex_attr *attr: Mutex 속성 값. 기본 특징을 이용시, NULL

  - pthread_mutex_lock()
    - Mutex lock을 얻기 위해 사용
    - 이미 다른 thread가 mutex lock을 얻고 있다면 얻을 때 까지 대기함

```
#include <pthread.h>

int pthread_mutex_lock(pthread_mutex_t *mutex);
```

      - pthread_mutex_t * mutex : lock 시킬 mutex 변수의 주소 값

Mutex: Mutual Exclusion (6/6)

- **APIs (cont'd)**
  - pthread_mutex_unlock()
    - Mutex lock을 반환하고자 할 때 사용

```
#include <pthread.h>

int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

      - pthread_mutex_t * mutex : unlock 시킬 mutex 변수의 주소 값

  - pthread_mutex_destroy()
    - Mutex객체를 제거하기 위해서 사용

```
#include <pthread.h>

int pthread_mutex_destroy(pthread_mutex_t *mutex);
```

      - pthread_mutex_t * mutex : 제거할 mutex 변수의 주소 값

실습1. Mutex: Lab 1(1/2)

- "MUTEX"가 정의되어 있으면 mutex를 사용한다는 의미

- 이는 컴파일시 "**-DMUTEX**"로 on 할 수 있음

```c
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>

void* thread_inc(void* arg);
void* thread_dec(void* arg);

#ifdef MUTEX
pthread_mutex_t mutex;
#endif

int a;
int num=0;

int main()
{
        int state;
        pthread_t t1, t2;
        void* thread_result1, *thread_result2;
#ifdef MUTEX
        state = pthread_mutex_init(&mutex, NULL);
        if(state)
        {
                printf("mutex init error!\n");
                return 1;
        }
#endif
        pthread_create(&t1, NULL, thread_inc, "thread1");
        pthread_create(&t2, NULL, thread_dec, "thread2");
        pthread_join(t1, &thread_result1);
        pthread_join(t2, &thread_result2);

        printf("[main] %d\n", num);
#ifdef MUTEX
        pthread_mutex_destroy(&mutex);
#endif
        return 0;
}
```

```c
void* thread_inc(void* arg)
{
        int i,j = 10000;
        for(i=0; i<10; i++)
        {
#ifdef MUTEX
                pthread_mutex_lock(&mutex);
#endif
                printf("[%s] %d\n", (char*)arg, num);
                a = i;
                while(--j);
                j = 10000;
                num += a;
                printf("[%s] %d\n", (char*)arg, num);
#ifdef MUTEX
                pthread_mutex_unlock(&mutex);
#endif
        }
        return (void*)&num;
}

void* thread_dec(void* arg)
{
        int i,j = 10000;
        for(i=0; i<5; i++)
        {
#ifdef MUTEX
                pthread_mutex_lock(&mutex);
#endif
                printf("[%s] %d\n", (char*)arg, num);
                a = i + 1;
                while(--j);
                j=10000;
                num -= a;
                printf("[%s] %d\n", (char*)arg, num);
#ifdef MUTEX
                pthread_mutex_unlock(&mutex);
#endif
        }
        return (void*)&num;
}
```

실습1. Mutex: Lab 1(2/2)

- Makefile

```
default:
	gcc -pthread -o mutex mutex.c
mutex:
	gcc -DMUTEX -pthread -o mutex mutex.c
clean:
	rm mutex
```

- 결과화면

```
sslab@sslab-VirtualBox:~$ make mutex
gcc -DMUTEX -pthread -o mutex mutex.c
sslab@sslab-VirtualBox:~$ ./mutex
[thread1] 0
[thread1] 0
[thread1] 0
[thread1] 1
[thread1] 1
[thread1] 3
[thread1] 3
[thread1] 6
[thread1] 6
[thread1] 10
[thread1] 10
[thread1] 15
[thread1] 15
[thread1] 21
[thread1] 21
[thread1] 28
[thread1] 28
[thread1] 36
[thread1] 36
[thread1] 45
[thread2] 45
[thread2] 44
[thread2] 44
[thread2] 42
[thread2] 42
[thread2] 39
[thread2] 39
[thread2] 35
[thread2] 35
[thread2] 30
[main] 30
sslab@sslab-VirtualBox:~$
```

KWANGWOON UNIVERSITY

2022년 1학기 **시스템프로그래밍실습** 14주차

4차퀴즈

System Software Laboratory
College of Software and Convergence
Kwangwoon Univ.

Qiuz

- 일시 : 15주차 수업시간, 비대면으로 진행
- 범위 : 12 ~ 14주차 강의자료, Proxy#3-1 ~ 3-2 과제
- 오픈북(실습강의자료만 가능)
- Klas 내 수시퀴즈 및 Zoom
  - 15분 동안 퀴즈 응시

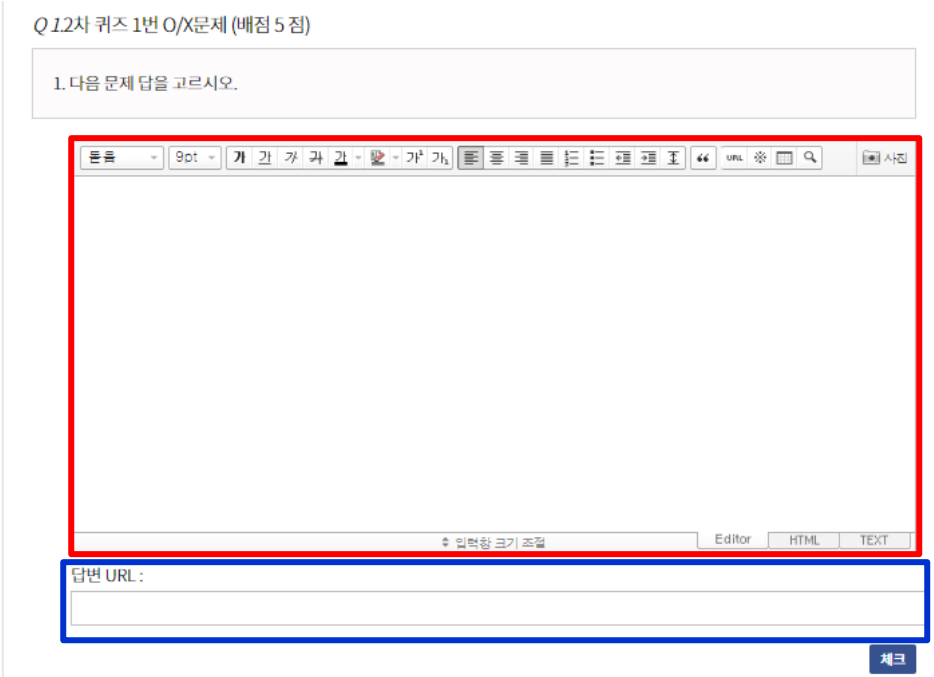
- 수강 요일 별 퀴즈 응시 기간
  - 목요일 7-8교시 수강하는 경우
    - 6월 9일 목요일 PM 6 : 00 ~ 6 : 30
  - 금요일 1-2교시 수강하는 경우
    - 6월 10일 금요일 AM 9 : 00~ 9 : 30
  - 금요일 5-6교시 수강하는 경우
    - 6월 10일 금요일 PM 3 : 00 ~ 3 : 30
  - **기본적으로 수강하는 요일 시간에 응시**
  - 수강하는 요일에 퀴즈를 응시 못하는 학생은 다른 요일 시간에 응시 가능
    - **mini896@naver.com or hj.cha@kw.ac.kr 중 메일로 응시 가능한 요일 작성 후 회신**
    - **회신해야 다른 요일에 퀴즈 응시 가능**

Qiuz

- 퀴즈 진행 방법
  - Klas 내 수시 퀴즈에서 답안만 작성(답안지)
    - 추후 공지사항에 암호화 된 docx 또는 pdf 형식의 시험지 공유 할 예정
  - 암호화 시험지 비밀번호
    - Zoom 내에 비밀번호를 공유할 예정
  - Zoom 입장
    - 퀴즈 응시 시간 기준으로 10분 까지만 Zoom 입장이 가능
      - ex) 목요일 인 경우 ➔ PM 6 : 00 ~ 6 : 10 까지만 입장이 가능
  - 퀴즈 관련 QnA Zoom 채팅 창 또는 음성으로 질문 가능
  - 퀴즈 응시 완료 후 Zoom 자율 퇴실
  - 추후 공지 사항에 Zoom링크 관련해서 공유할 예정

Qiuz

- 퀴즈 답안 작성 안내
  - 답안 작성 시 답안 공간(빨간 박스)과 답변 URL(파란 박스) 두 군데에 답 표기



  - 서버 문제로 인해 퀴즈 응시 완료 후 작성한 답안들이 사라지는 경우가 있어 두 군데에 전부 답 표기를 해야 함.