Operating System 실습
[Synchronization]

한양대학교
HANYANG UNIVERSITY

Data-Centric Systems Lab.

Contact

- Instructor

  - Name : 최기한
  - Tel. : 02-2220-4439
  - E-Mail : choikihan1@hanyang.ac.kr
  - Office : 산학기술관 614호

Kernel Preemption

- Synchronization Primitives
  - Various types of synchronization techniques used by the kernel

Technique	Description	Scope
Per-CPU variables	Duplicate a data structure among the CPUs	All CPUs
Atomic operation	Atomic read-modify-write instruction to a counter	All CPUs
Memory barrier	Avoid instruction reordering	Local CPU or All CPUs
Spin lock	Lock with busy wait	All CPUs
Semaphore	Lock with blocking wait (sleep)	All CPUs
Seqlocks	Lock based on an access counter	All CPUs
Local interrupt disabling	Forbid interrupt handling on a single CPU	Local CPU
Local softirq disabling	Forbid deferrable function handling on a single CPU	Local CPU
Read-copy-update (RCU)	Lock-free access to shared data structures through pointers	All CPUs

Mutex (pthread)

- Header File
  - #include <pthread.h>

- Mutex 초기화
  - int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *mutexattr);
  - Mutex :초기화 하고자 하는 뮤텍스의 포인터
  - Mutexattr : 뮤텍스의 특성(attribute)을 설정할 때 사용 (사용 안할시 NULL)

- Mutex lock
  - int pthread_mutex_lock(pthread_mutex_t *mutex);
  - 뮤텍스 잠금 요청(락되어있다면 락이 풀릴때까지 대기, 락이 않되었다면 락을하고 임계영역 진입)

- Mutex unlock
  - int pthread_mutex_unlock(pthread_mutex_t *mutex);
  - 뮤텍스 잠금을 해제(락을 푼다.)

- Mutex 제거
  - int pthread_mutex_destroy(pthread_mutex_t *mutex);
  - 사용하고 난 뒤 뮤텍스와 이에 관련된 리소스를 해제

- Return value : 모든 함수 성공 시 0을 리턴한다.

실습 (Mutex)

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

int num=0;
pthread_mutex_t mutex;

void *thread_func(void *arg);
void Error(const char *mes);

int main()
{
        pthread_t t1, t2;
        char *thread1 = "Thread_A";
        char *thread2 = "Thread_B";
        void *t_return = NULL;

        if(pthread_mutex_init(&mutex,NULL) !=0)
                Error("pthread_mutex_init Error\n");

        pthread_create(&t1, NULL, thread_func, thread1);
        pthread_create(&t2, NULL, thread_func, thread2);

        pthread_join(t1, &t_return);
        pthread_join(t2, &t_return);

        if(pthread_mutex_destroy(&mutex) !=0)
                Error("pthread_mutext_destory Error\n");
        printf("Finale Number : %d\n",num);

        return 0;
}
```

```c
void *thread_func(void *arg)
{
        int i=0;
        for(i=0;i<5;i++)
        {
                sleep(1);
                pthread_mutex_lock(&mutex);
                sleep(1);
                num++;
                printf("%s\t:%d\n",(char*)arg,num);
                pthread_mutex_unlock(&mutex);
        }
}

void Error(const char *mes)
{
        printf("%s\n", mes);
        exit(0);
}
```

Semaphore (pthread)

- Header File
  - #include <Semaphore.h>

- Semaphore 초기화
  - int sem_init(sem_t *sem, int pshared, unsigned int value);

- Semaphore value 감소
  - int sem_wait(sem_t *sem);
  - int sem_trywait(sem_t *sem);
    - * 0에서 대기하지 않고 바로 EAGAIN(error의 일종) 를 리턴 (Non blocking)

- Semaphore value 증가
  - int sem_post(sem_t *sem);

- Semaphore value 저장
  - int sem_getvalue(sem_t *sem, int *sval);
  - sval 이 가리키는 위치에 sem 세마포어의 현재값을 저장

- Semaphore 삭제 (할당된 자원 해제)
  - int sem_destroy(sem_t *sem);

Semaphore (pthread)

- Return value
  - sem_wait(), sem_getvalue() : 항상 0을 반환
  - 다른 모든 세마포어 함수들은 성공시 0, 실패시 -1 반환
  - 실패 반환시 errno변수에 해당 에러코드를 기록

- Error
  - sem_init()
    - EINVAL : value는 카운터의 최대값인 SEM_VALUE_MAX 값보다 크다
    - ENOSYS : pshared 값이 0이 아니다.
  - sem_trywait()
    - EAGAIN : 현재 세마포어의 카운터 값이 0이다.
  - sem_post()
    - ERANGE : 증가 후 세마포어의 값이 SEM_VALUE_MAX 값보다 커진다.
  - sem_destroy()
    - 다른 쓰레드가현재 해당 세마포어를 기다리는 중이다.

실습 (Semaphore)

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

void * thread_snd(void *arg);
void * thread_rcv(void *arg);

sem_t bin_sem;

char thread1[] = "A Thread(SND)";
char thread2[] = "B Thread(RCV)";
char thread3[] = "C Thread(RCV)";

int main(int argc, char *argv[])
{
        pthread_t t1, t2, t3;
        void *thread_result;
        int sem_value;
        int state;

        state = sem_init(&bin_sem, 0, 0);
        if(state != 0)
        {
                puts("sem_init() error");
                exit(1);
        }

        state = pthread_create(&t1, NULL, thread_snd, (void *)&thread1);
        state = pthread_create(&t2, NULL, thread_rcv, (void *)&thread2);
        state = pthread_create(&t3, NULL, thread_rcv, (void *)&thread3);

        pthread_join(t1, &thread_result);
        pthread_join(t2, &thread_result);
        pthread_join(t3, &thread_result);

        sem_getvalue(&bin_sem, &sem_value);
        printf("sem_getvalue : %d\n", sem_value);
        sem_destroy(&bin_sem);

        return EXIT_SUCCESS;

}
```

실습 (Semaphore)

```c
void *thread_snd(void* arg)
{
        int i;
        int sema_value;

        for(i=0; i<4; i++)
        {
                do{
                        sem_getvalue(&bin_sem, &sema_value);
                }while(sema_value != 0);

                sem_getvalue(&bin_sem, &sema_value);
                printf("Execution : %s, sem_getvalue : %d \n", (char *)arg, sema_value+1);
                sem_post(&bin_sem);
        }
}

void* thread_rcv(void* arg)
{
        int i;
        int sema_value;

        for(i=0;i<2;i++)
        {
                sleep(1);
                sem_wait(&bin_sem);
                sem_getvalue(&bin_sem, &sema_value);
                printf("Execution : %s, sem_getvalue : %d \n", (char *)arg, sema_value);
        }
}
```