인덕대학교

교착 및 기아 상태
- 임계 구역

컴퓨터소프트웨어학과

김병국 교수

학습목표

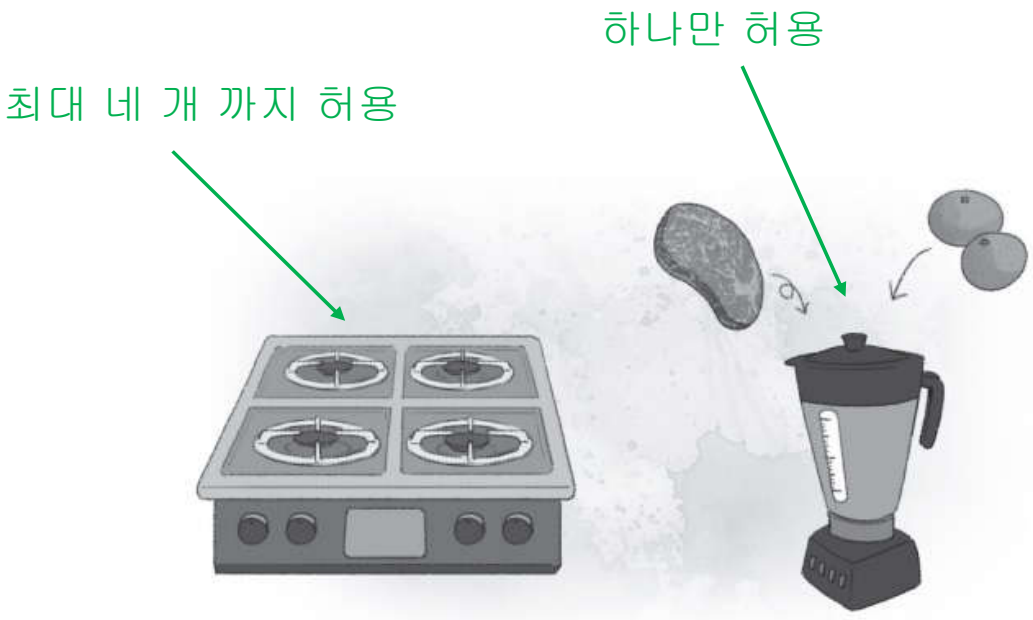
- 임계 구역에 대하여 정의할 수 있다.
- 간단한 스레드 프로그래밍을 할 수 있다.
- 임계 구역을 해결하기 위한 세마포어 기법을 사용할 수 있다.

목차

1. 임계 구역 (1/3)

❑ Critical Area

- 또는 “**임계 영역**”이라 함
- 공유 자원에 대하여 프로세스(멀티스레드 포함)들의 <u>동시 접근에 한계가 있는 영역</u>
- 임계 구역을 접근할 때에는 반드시 <u>가용 상태를 확인</u>해야 함

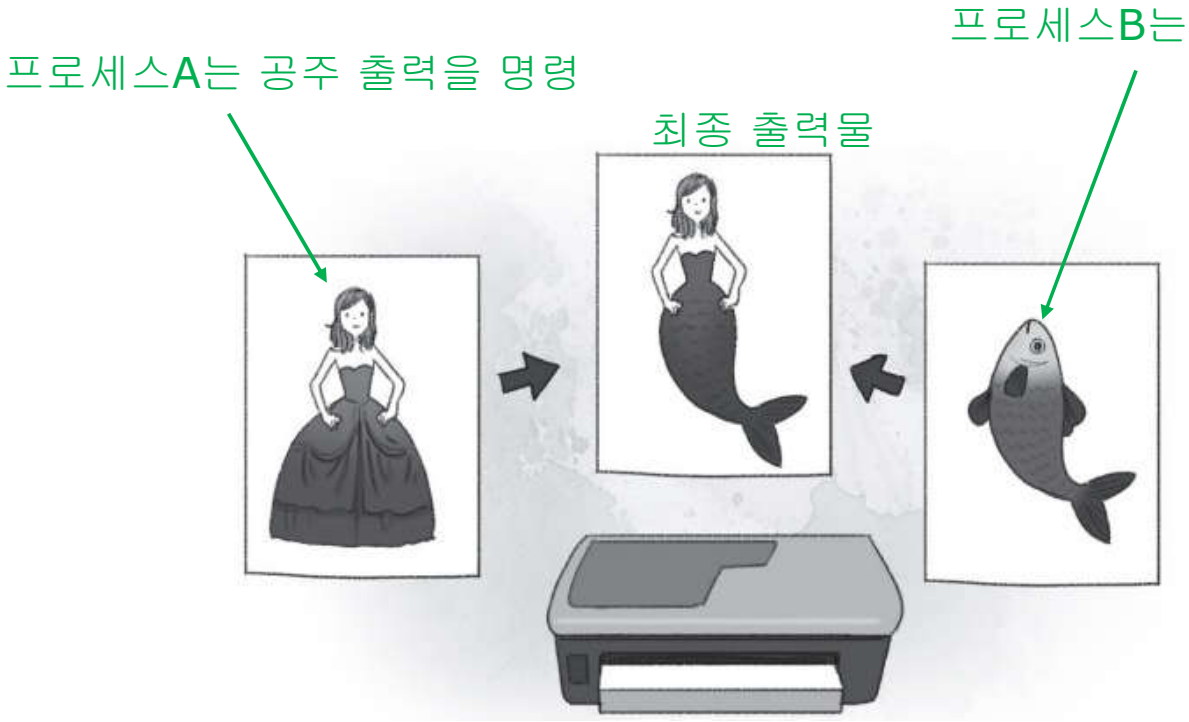


[가스레이지와 믹서]

1. 임계 구역 (2/3)

❑프린터 예시

- 만약 프린터장치가 프로세스들의 동시 접근을 허용하게 된다면,
- 프로세스들에 의한 출력 요청 명령 들로 인해 기대 외의 결과물이 나오게 됨

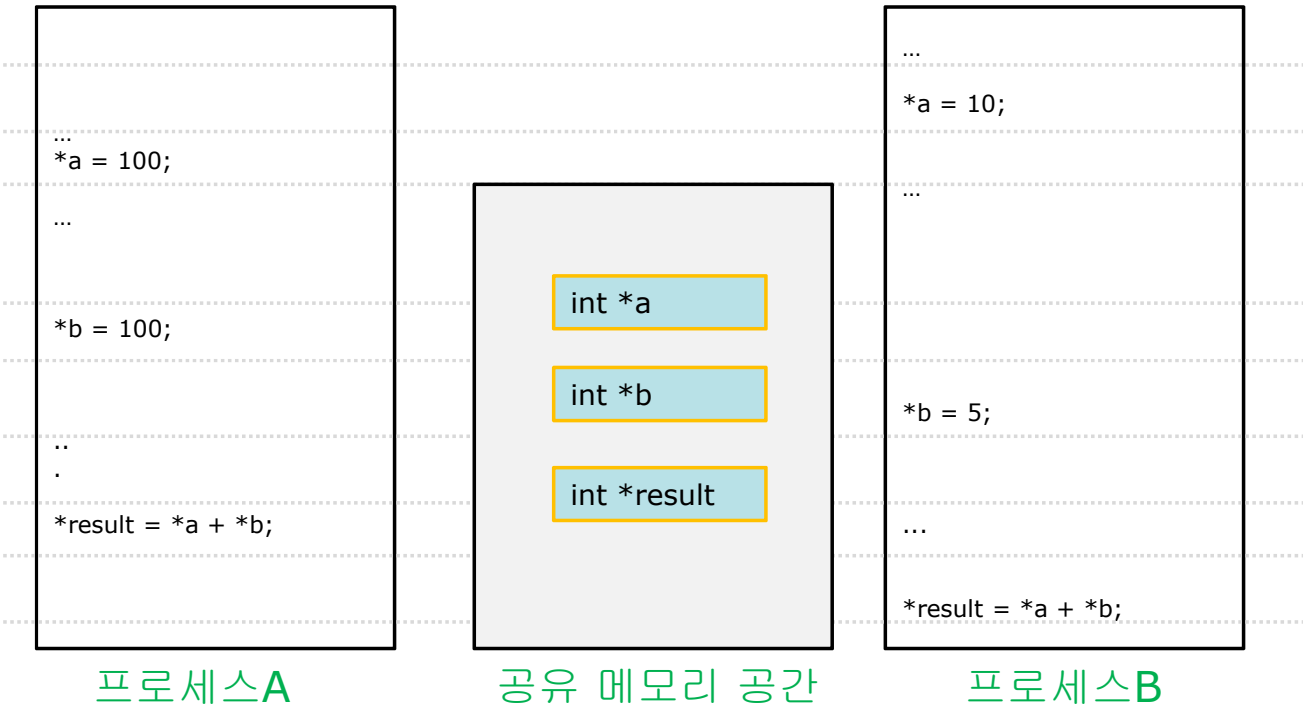


[프린터 공유 문제]

1. 임계 구역 (3/3)

❑ 공유 메모리 사용 예시

- *result의 값은?



2. 임계 구역 해결 조건

❑상호 배제(mutual exclusion)

- 한 프로세스가 임계 구역에 들어가면 다른 프로세스는 그곳에 들어갈 수 없음

❑한정 대기(bounded waiting)

- 어떤 프로세스도 무한 대기하지 않아야 함

❑진행의 융통성(progress flexibility)

- 다른 프로세스의 진행을 방해해서는 안 됨

3. 실습 1 (1/4)

❑ 임계 구역 데이터의 동시 접근 실습 (1/4)

- 코드 작성

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

void* mythread_r(void* ptr);
void* mythread_w(void* ptr);

int
main(int argc, char* argv[])
{
    pthread_t thread1, thread2;
    int nValue = 0;

    // start the threads
    pthread_create(&thread1, NULL, *mythread_r, (void*)&nValue);
    pthread_create(&thread2, NULL, *mythread_w, (void*)&nValue);

    // wait for threads to finish
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    return 0;
}
```

3. 실습 1 (2/4)

❑ 임계 구역 데이터의 동시 접근 실습 (2/4)

- 코드 작성

```
29  void*
30  mythread_r(void* p_Value)
31  {
32      int* p_nValue = (int*)p_Value;
33
34      do
35      {
36          if (*p_nValue % 2 == 0)
37          {
38              usleep(500000);
39              printf("%d is even number!!\n", *p_nValue);
40          }
41          else
42          {
43              usleep(500000);
44              printf("%d is odd number!!\n", *p_nValue);
45          }
46
47          usleep(500000);
48      } while (1);
49
50      return  p_nValue;
51  }
52
```

```
53  void*
54  mythread_w(void* p_Value)
55  {
56      int* p_nValue = (int*)p_Value;
57
58      do
59      {
60          (*p_nValue)++;
61
62          usleep(500000);
63      } while (1);
64
65      return  p_nValue;
66  }
```

3. 실습 1 (3/4)

❑ 임계 구역 데이터의 동시 접근 실습 (3/4)

- 코드 빌드 및 실행
- 주의 스레드 관련 빌드를 위해서는 반드시 "-lpthread" 옵션을 gcc명령에 함께 사용해야 함

```
[kali@kali:05_1]$gcc critical_section.c -lpthread
[kali@kali:05_1]$./a.out
2 is odd number!!
4 is odd number!!
6 is odd number!!
8 is odd number!!
9 is even number!!
11 is even number!!
13 is even number!!
15 is even number!!
17 is even number!!
19 is even number!!
22 is even number!!
23 is odd number!!
26 is odd number!!
27 is odd number!!
30 is odd number!!
^C
[kali@kali:05_1]$
```

빌드

대체적으로 우리가 원하는 값이 맞음?

3. 실습 1 (4/4)

❑임계 구역 데이터의 동시 접근 실습 (4/4)

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

void* mythread_r(void* ptr);
void* mythread_w(void* ptr);
```

1

```c
int
main(int argc, char* argv[])
{
    pthread_t thread1, thread2;
    int nValue = 0;

    // start the threads
    pthread_create(&thread1, NULL, *mythread_r, (void*)&nValue);
    pthread_create(&thread2, NULL, *mythread_w, (void*)&nValue);

    // wait for threads to finish
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    return 0;
}
```

2

```c
void*
mythread_r(void* p_Value)
{
    int* p_nValue = (int*)p_Value;

    do
    {
        if (*p_nValue % 2 == 0)
        {
            usleep(500000);
            printf("%d is even number!!\n", *p_nValue);
        }
        else
        {
            usleep(500000);
            printf("%d is odd number!!\n", *p_nValue);
        }

        usleep(500000);
    } while (1);

    return  p_nValue;
}
```

3

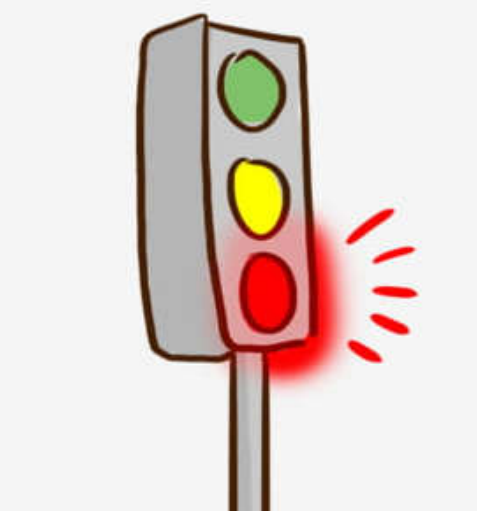
```c
void*
mythread_w(void* p_Value)
{
    int* p_nValue = (int*)p_Value;

    do
    {
        (*p_nValue)++;

        usleep(500000);
    } while (1);

    return  p_nValue;
}
```

4

4. 세마포어

❑ Semaphore

- 임계 구역 접근을 위해 On/Off 표시등(lamp)을 운영
  - On 상태 : 사용 중
  - Off 상태 : 사용 가능
- 임계 구역을 접근하는 프로세스들은 이 표시등을 모니터링(반복 수행)
- 가용상태(Off 상태)이면, 자신이 On시키고 접근



5. 실습 2 (1/3)

❑ 세마포어 적용 후 실습 1의 문제 해결 (1/3)

- 코드 작성

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

void* mythread_r(void* ptr);
void* mythread_w(void* ptr);

sem_t g_mutex;

int
main(int argc, char* argv[])
{
    pthread_t thread1, thread2;
    int nValue = 0;

    sem_init(&g_mutex, 0, 1); // initialize g_mutex

    // start the threads
    pthread_create(&thread1, NULL, *mythread_r, (void*)&nValue);
    pthread_create(&thread2, NULL, *mythread_w, (void*)&nValue);

    // wait for threads to finish
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    sem_destroy(&g_mutex); // destroy g_mutex

    return 0;
}
```

코드 추가

5. 실습 2 (2/3)

❑ 세마포어 적용 후 실습 1의 문제 해결 (2/3)

- 코드 작성

```
36    void*
37    mythread_r(void* p_Value)
38    {
39        int* p_nValue = (int*)p_Value;
40
41        do
42        {
43            sem_wait(&g_mutex); // wait and lock g_mutex
44            if (*p_nValue % 2 == 0)
45            {
46                usleep(500000);
47                printf("%d is even number!!\n", *p_nValue);
48            }
49            else
50            {
51                usleep(500000);
52                printf("%d is odd number!!\n", *p_nValue);
53            }
54            sem_post(&g_mutex); // release g_mutex
55
56            usleep(500000);
57        } while (1);
58
59        return  p_nValue;
60    }
```

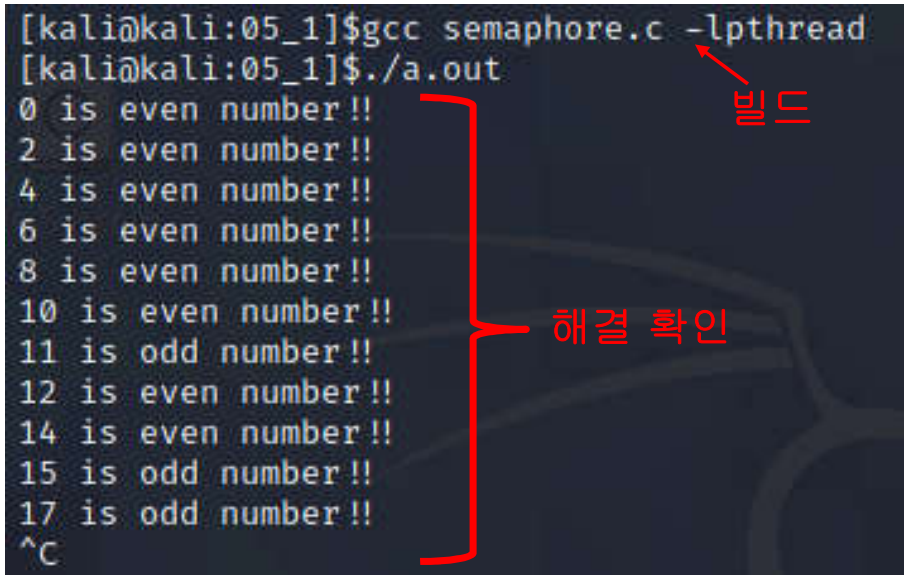
코드 추가

```
62    void*
63    mythread_w(void* p_Value)
64    {
65        int* p_nValue = (int*)p_Value;
66
67        do
68        {
69            sem_wait(&g_mutex);// wait and lock g_mutex
70            (*p_nValue)++;
71            sem_post(&g_mutex); // release g_mutex
72
73            usleep(500000);
74        } while (1);
75
76        return  p_nValue;
77    }
```

5. 실습 2 (3/3)

❑ 세마포어 적용 후 실습 1의 문제 해결 (3/3)

- 빌드 및 실행



6. 모니터

❑ Monitor

- 공유 자원을 내부로 숨김
- 외부 접속을 차단
- 인터페이스를 통해 공유 자원 갱신 및 데이터 추출

❑ 작동 원리

- 요청받은 작업을 모니터 큐에 저장
- 시스템은 모니터 큐의 데이터를 순서대로 처리
- 결과만 해당 프로세스에 알려줌

수고하셨습니다.