인덕대학교

메모리 관리
- 기본 개요 및 공유 메모리

컴퓨터소프트웨어학과

김병국 교수

학습목표

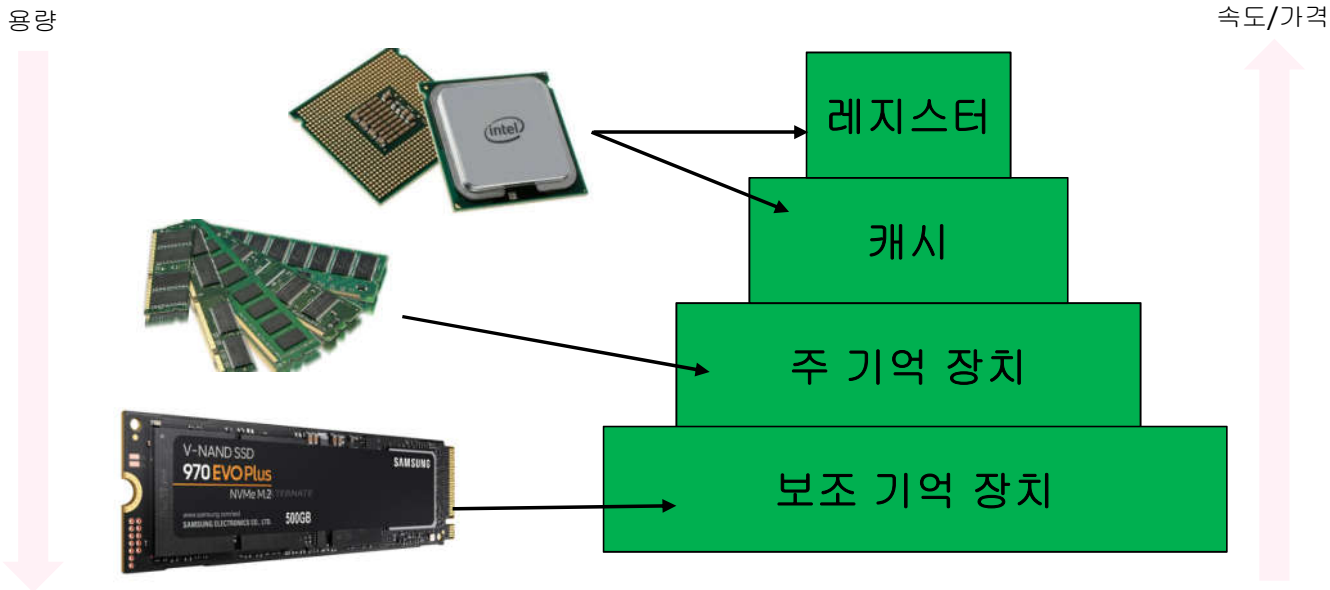
- 운영체제의 메모리 관리의 필요성을 이해한다.
- 시스템에서 메모리 공간의 이용 상태를 확인할 수 있다.
- 프로세스간 메모리를 공유하는 방법을 안다.

목차

1. 메모리 관리 개요 (1/4)

❑ 메모리의 계층화

- CPU 연산결과가 기록되는 레지스터의 경우 빠른 입/출력이 가능
- 메모리의 읽고/쓰기의 속도는 가격에 비례
- 효율적 메모리 사용을 위해 속도와 용량의 관계를 두고 계층화

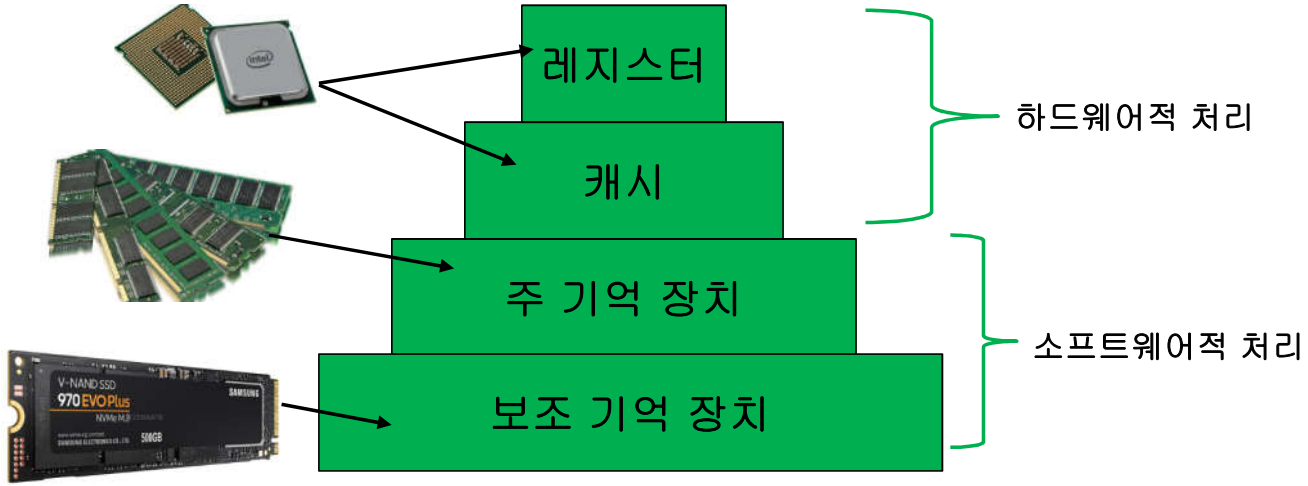


1. 메모리 관리 개요 (2/4)

인덕대학교

❑ 메모리의 제어

- 레지스터와 캐시는 CPU 제조사에 의해 접근 방식이나 데이터의 기록 방식이 하드웨어적으로 구현됨
- 주기억 장치와 보조 기억 장치의 데이터 구조 및 저장 방식은 소프트웨어적으로 구현됨
  - 주 기억 장치 → 운영체제의 메모리 관리
  - 보조 기억 장치의 저장 방식 → 파일 시스템, 가상메모리

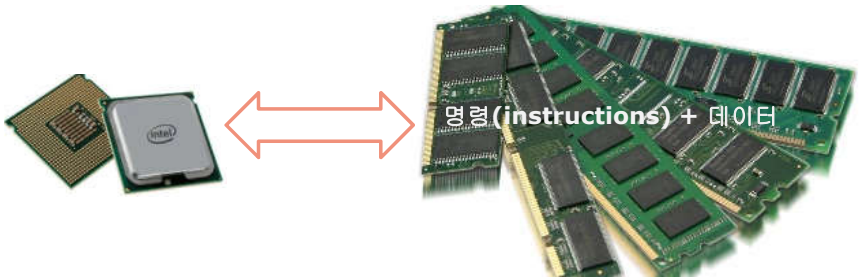


1. 메모리 관리 개요 (3/4)

❑ 프로세스를 위한 메모리 접근

- 프로세스
  - 메모리에 적재되어 주기적으로 CPU를 점유하여 실행 상태에 진입이 가능한 것
  - 소위, 메모리에 명령들(instructions set)이 존재

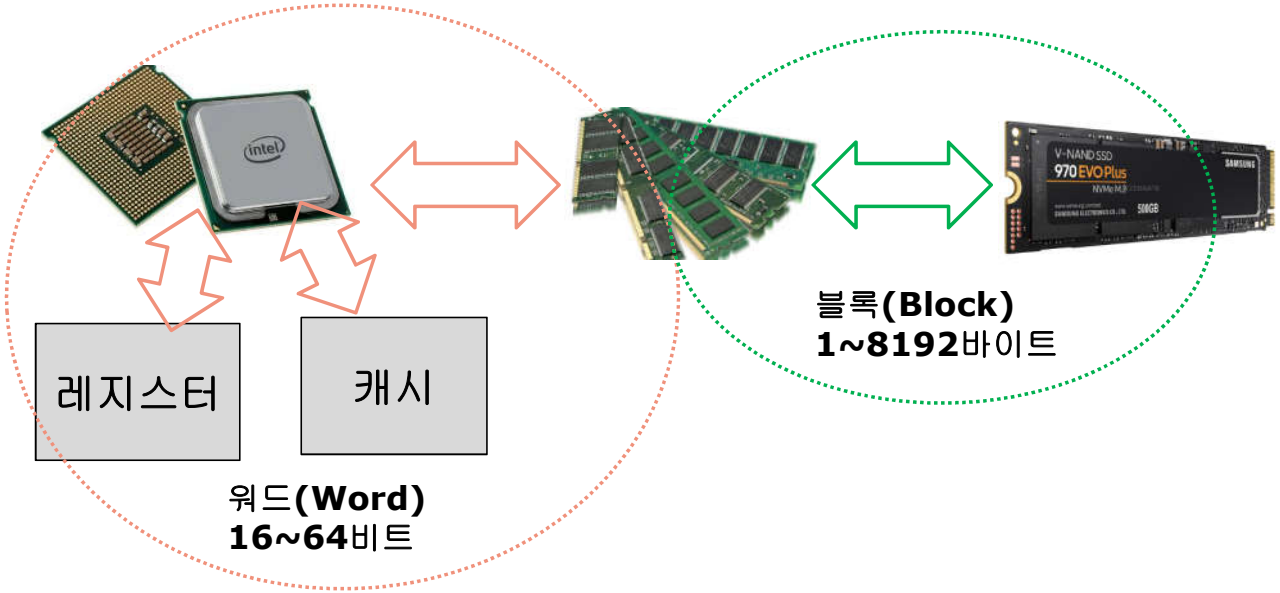
- 메모리
  - 큰 바이트 순열(bytes array)로 구성
  - 바이트별 주소(address)가 있음
  - CPU는 PC(program counter) 레지스터의 값에 해당하는 메모리 주소에 접근하여 명령들을 가져옴
    - 명령 처리에 의해 메모리에 읽기/쓰기를 수행



1. 메모리 관리 개요 (4/4)

❑ 데이터 접근 단위

- CPU와 직접 연결되는 메모리들은 워드(Word) 단위로 접근
- 보조 기억장치의 데이터는 파일시스템을 구성할 때 설정한 블록(Block) 크기의 단위로 접근



2. 메모리 이용 확인 (1/2)

❑ 메모리 관련 리눅스 명령

- 명령어: free
  - 메모리의 이용 상태(usage)를 보여줌
  - 옵션:
    - -t : 모든(total) 용량을 출력
    - -h : 단위 문자 출력
    - -g : gigabyte 단위 출력
    - -m : megabyte 단위 출력
    - -k : kilobyte 단위 출력
    - -b : byte 단위 출력

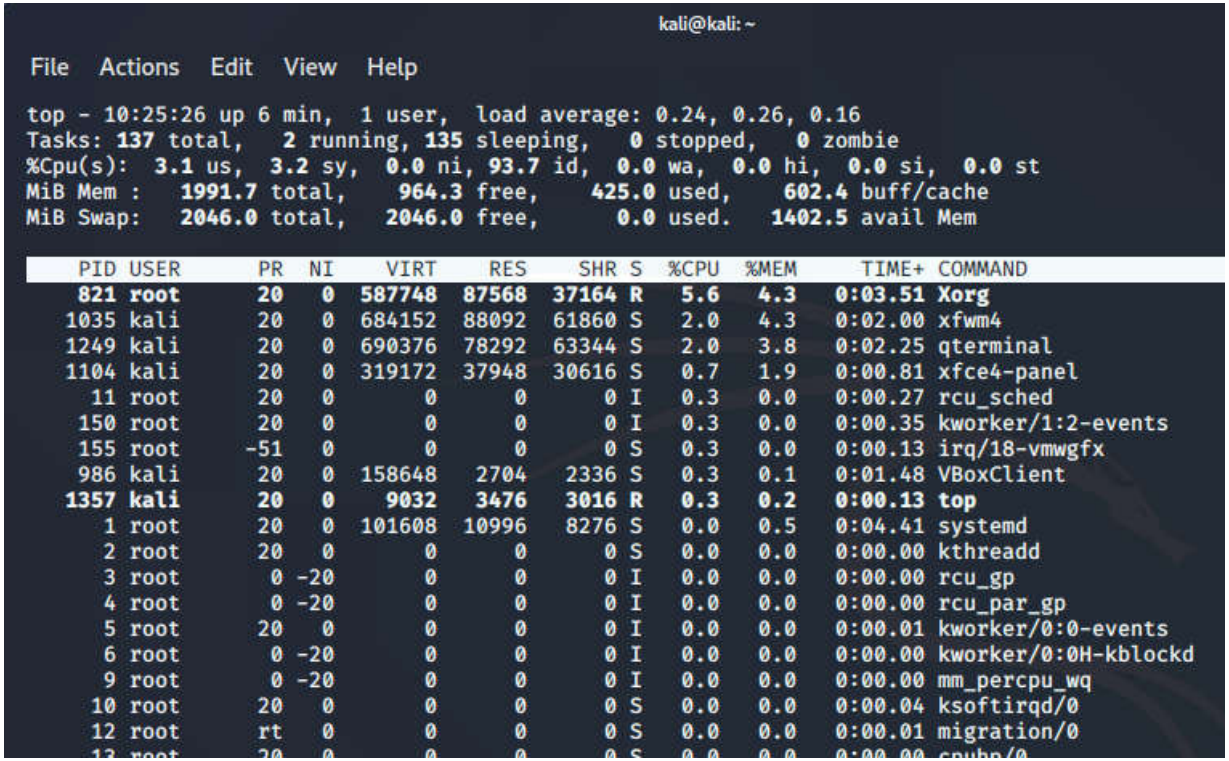
```
kali@kali:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           1.9Gi       423Mi       966Mi        11Mi       601Mi       1.4Gi
Swap:          2.0Gi          0B       2.0Gi
kali@kali:~$ free -h -t
               total        used        free      shared  buff/cache   available
Mem:           1.9Gi       423Mi       966Mi        11Mi       601Mi       1.4Gi
Swap:          2.0Gi          0B       2.0Gi
Total:         3.9Gi       423Mi       2.9Gi
kali@kali:~$
```

[free명령어 동작 예]

2. 메모리 이용 확인 (2/2)

❑메모리 관련 리눅스 명령

- 명령어: top



[top명령어 동작 예]

2. 메모리 이용 확인 (2/2)

❑ 메모리 점유상태 확인

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

int
main(int argc, char* argv[])
{
    int nCount = atoi(argv[1]);
    char *p = 0;

    printf("Allocating Memories....\n");
    for(int i=0; i<nCount; i++)
    {
        p = (char*)malloc(4096);
    }

    printf("Done....\n");
    getchar();

    return 0;
}
```

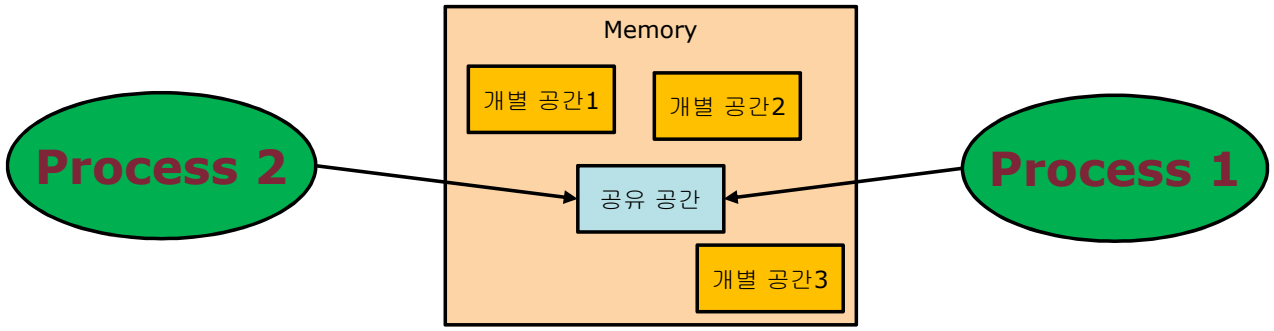
```
[kali@kali:06_1]$free -m
               total        used        free      shared  buff/cache   available
Mem:            3932         875        2234          15         822        2797
Swap:            974           0         974
[kali@kali:06_1]$ps au
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         591  0.0  2.8 1204316 114216 tty7   Ssl+ Mar29   0:24 /usr/lib/xorg/Xorg
root         592  0.0  0.0   5784  1772 tty1     Ss+  Mar29   0:00 /sbin/agetty -o -p
kali        1220  0.0  0.1   8736  5720 pts/0    Ss   Mar29   0:00 /usr/bin/bash
kali        1284  0.0  0.1   8876  5648 pts/1    Ss   Mar29   0:00 /usr/bin/bash
kali        1345  0.0  0.1   8872  5660 pts/2    Ss+  Mar29   0:00 /usr/bin/bash
kali        8190  0.7  9.9 403824 401044 pts/1   S+   09:12   0:00 ./malloc 100000
kali        8192  0.0  0.0   9636  3248 pts/0    R+   09:12   0:00 ps au
[kali@kali:06_1]$
```

```
Shell No.1
File  Actions  Edit  View  Help
[kali@kali:06_1]$gcc malloc.c  -o malloc
[kali@kali:06_1]$./malloc 100000
Allocating Memories....
Done....
```

3. 메모리 공유 (1/10)

❑공유 메모리(Shared Memory)

- **동일한 메모리공간**을 서로 다른 프로세스들이 공유
- 운영체제를 통해 **공간이 공유됨**



3. 메모리 공유 (2/10)

❑ 공유 메모리 사용 절차

1. 공유 메모리 생성
2. 프로세스에 공유 영역을 첨부
3. 공유 영역 접근
4. 첨부된 공유 영역을 해제
5. 공유 메모리 삭제

3. 메모리 공유 (3/10)

❑ 공유 메모리 생성 (1/2)

- 공유 메모리 생성
- 인자
  - key : 시스템에서 식별하기 위한 공유메모리 번호
  - size: 공유메모리 크기
  - shmflg: 동작 옵션
    - IPC_CREAT :
      - » key에 해당하는 공유메모리가 없으면 생성(단, 생성 시 접근 권한을 부여해야 함)
      - » 동일 값이 이미 있으면, 무시됨
    - IPC_EXCL :
      - » 공유메모리가 이미 있으면 실패 의미로 -1을 반환
- 반환값:
  - 성공: 공유메모리 식별자
  - 실패: -1

```
#include <sys/ipc.h>
#include <sys/shm.h>

int shmget(key_t key, size_t size, int shmflg);
```

[shmget 함수 프로토타입]

3. 메모리 공유 (4/10)

❑ 공유 메모리 생성 (2/2)

```c
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>

int
main()
{
    int nShmId=0;

    nShmId = shmget(0x1234, 4096, IPC_CREAT|IPC_EXCL|0666);
    printf("Shared Memory ID is %d\n", nShmId);

    return 0;
}
```

```
[kali@kali:06_1]$
[kali@kali:06_1]$gcc shmget.c
[kali@kali:06_1]$./a.out
Shared Memory ID is 32796
[kali@kali:06_1]$./a.out
Shared Memory ID is -1
[kali@kali:06_1]$
```

생성 성공

생성 실패

```
[kali@kali:06_1]$ipcs -m

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000000 32776      kali       600        524288     2          dest
0x00000000 32778      kali       600        2097152    2          dest
0x00000000 14         kali       600        524288     2          dest
0x00000000 15         kali       600        2097152    2          dest
0x00000000 16         kali       600        524288     2          dest
0x00000000 19         kali       600        524288     2          dest
0x00000000 32789      kali       600        524288     2          dest
0x00000000 26         kali       600        524288     2          dest
0x00001234 32796      kali       666        4096       0
0x00000000 29         kali       600        524288     2          dest
0x00000000 32         kali       600        524288     2          dest
0x00000000 37         kali       600        524288     2          dest
0x00000000 50         kali       600        524288     2          dest

[kali@kali:06_1]$
```

3. 메모리 공유 (5/10)

❑ 공유 메모리 접근

- 공유메모리를 프로세스 메모리에 첨부 (attach)
- 인자:
  - shmid : 공유메모리 식별자
  - shmaddr : 공유메모리 주소(일반적으로 NULL을 사용)
  - shmflg : 동작옵션
- 반환값
  - 성공: 공유메모리 주소 포인터
  - 실패: (void *)-1

```
#include <sys/types.h>
#include <sys/shm.h>

void *shmat(int shmid, const void *shmaddr, int shmflg);
```

[shmat 함수 프로토타입]

3. 메모리 공유 (6/10)

❑ 공유 메모리 해제

- 공유메모리를 프로세스 메모리로부터 분리(detach)
- 모두 분리되었다고 해서 공유메모리가 사라지지는 않음
- 강제 소거를 위한 절차가 필요

```
#include <sys/types.h>
#include <sys/shm.h>

int shmdt(const void *shmaddr);
```

[shmdt 함수 프로토타입]

- 반환값:
  - 성공: 0
  - 실패: -1

3. 메모리 공유 (7/10)

❑공유 메모리 사용 예

- 코드: shm_w.c

```c
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <string.h>
```

1

```c
int
main()
{
    int nShmId=0;
    char *pShared=0;

    nShmId = shmget(0x1234, 4096, IPC_CREAT|0666);
    printf("Shared Memory ID is %d\n", nShmId);

    pShared = (char *)shmat(nShmId, NULL, 0);

    memset(pShared, 0, 4096);
```

2

```c
    for(int i=0; i<20; i++)
    {
        sprintf(pShared, "Shared : %d", i);
        sleep(1);
    }

    shmdt(pShared);

    return 0;
}
```

3

3. 메모리 공유 (8/10)

❑공유 메모리 사용 예

- 코드: shm_r.c

```c
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <string.h>
```

1

```c
int
main()
{
    int nShmId=0;
    char *pShared=0;

    nShmId = shmget(0x1234, 4096, IPC_CREAT|0666);
    printf("Shared Memory ID is %d\n", nShmId);

    pShared = (char *)shmat(nShmId, NULL, 0);

    memset(pShared, 0, 4096);
```

2

```c
    for(int i=0; i<20; i++)
    {
        printf("Data: %s\n", pShared);
        sleep(1);
    }

    shmdt(pShared);

    return 0;
}
```

3

3. 메모리 공유 (9/10)

❑ 공유 메모리 수정 및 삭제 (1/2)

- 생성된 공유메모리에 대하여 속성을 수정
- 공유메모리를 운영체제로부터 삭제
- 인자 :
  - shmid : 공유메모리 ID
  - cmd : 공유메모리 제어(삭제: IPC_RMID, 설정: IPC_SET)
  - buf : 속성값 추출 및 설정(삭제 시 NULL)
- 반환값:
  - 성공: 0
  - 실패: -1

```
struct shmid_ds {
    struct ipc_perm shm_perm;    /* Ownership and permissions */
    size_t          shm_segsz;   /* Size of segment (bytes) */
    time_t          shm_atime;   /* Last attach time */
    time_t          shm_dtime;   /* Last detach time */
    time_t          shm_ctime;   /* Creation time/time of last
                                    modification via shmctl() */
    pid_t           shm_cpid;    /* PID of creator */
    pid_t           shm_lpid;    /* PID of last shmat(2)/shmdt(2) */
    shmatt_t        shm_nattch;  /* No. of current attaches */
    ...
};
```

[buf에 대한 자료구조]

```
#include <sys/ipc.h>
#include <sys/shm.h>

int shmctl(int shmid, int cmd, struct shmid_ds *buf);
```

[shmctl 함수 프로토타입]

3. 메모리 공유 (10/10)

❑ 공유 메모리 수정 및 삭제 (2/2)

```c
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>
```

①

②

```c
int
main(int argc, char* argv[])
{
    int nShmId=0;

    if(argc != 2)
    {
        printf("We need one argument!!\n");
        return 0;
    }

    nShmId = atoi(argv[1]);
    printf("Shared Memory ID is %d\n", nShmId);

    if( shmctl(nShmId, IPC_RMID, NULL) )
    {
        printf("Deleting Shared Memory has been failed.\n");
        return -1;
    }

    printf("Shared Memory is deleted successfully\n");

    return 0;
}
```

[실행결과]

```
[kali@kali:06_1]$./shmget
Shared Memory ID is 32771
[kali@kali:06_1]$ipcs -m

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      natt
0x00000000 32768      kali       600        524288     2
0x00000000 32769      kali       600        2097152    2
0x00001234 32771      kali       666        4096       0
0x00000000 12         kali       600        524288     2

[kali@kali:06_1]$./shmctl 32771
Shared Memory ID is 32771
Shared Memory is deleted successfully
[kali@kali:06_1]$ipcs -m

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      natt
0x00000000 32768      kali       600        524288     2
0x00000000 32769      kali       600        2097152    2
0x00000000 12         kali       600        524288     2

[kali@kali:06_1]$
```

수고하셨습니다.