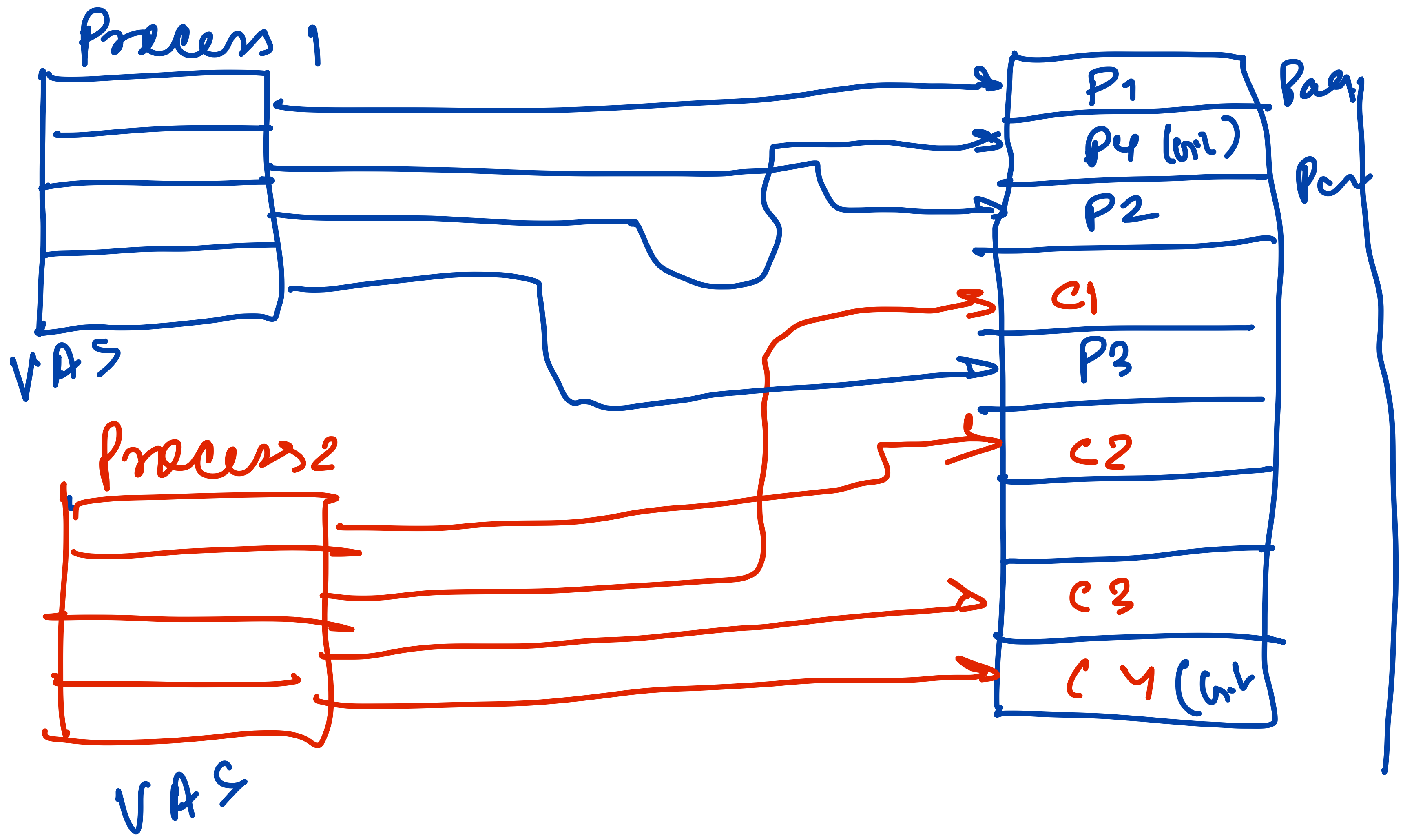
Operating Systems (CS3000)

Lecture – 12
(Inter Process Communication)

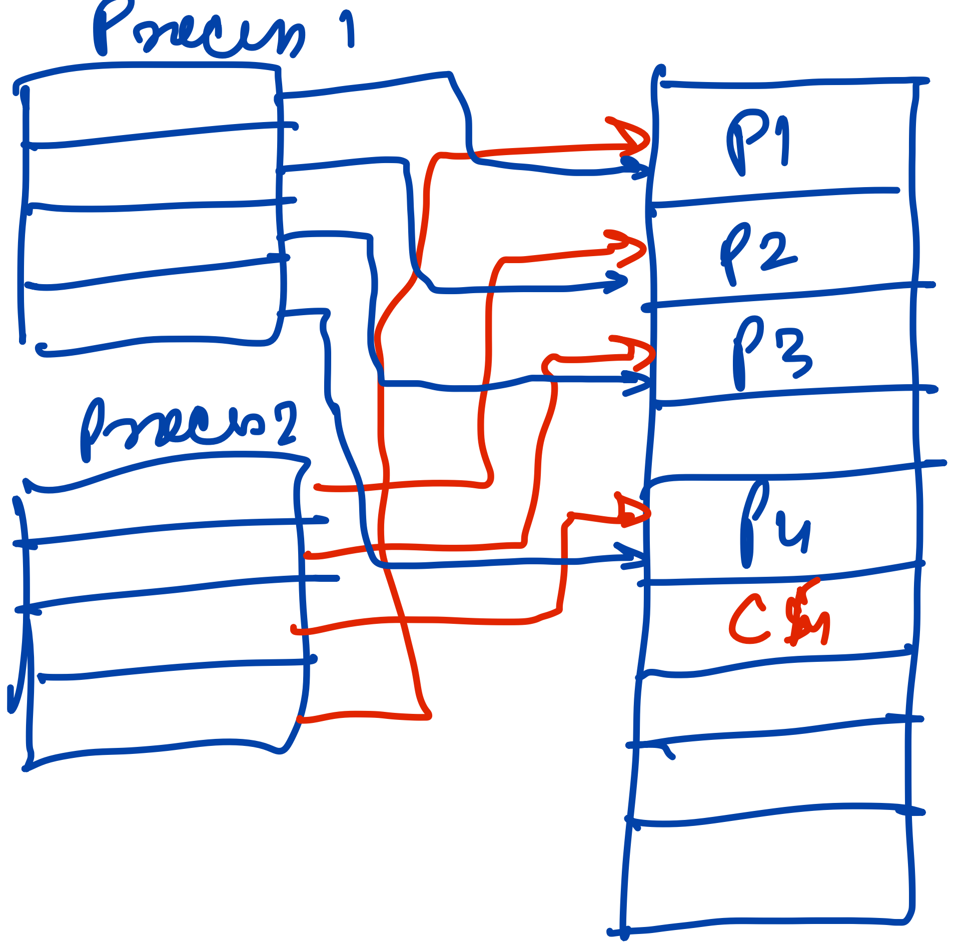
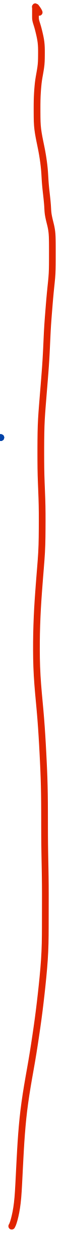
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING, KANCHEEPURAM

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING, KANCHEEPURAM

Dr. Jaishree Mayank
Assistant Professor
Department of Computer Sc. and Engg.



Process 1
VAS
Process2
VAS
P1
P4 (on 2)
P2
C1
P3
C2
C3
C4 (on 2)
Page
Pc


Process 1
Process 2
P1
P2
P3
P4
C5M


IPC

- Processes within a system may be **independent** or **cooperating**
- Cooperating process can affect or be affected by other processes
    - Same computer or networked computers

IPC

- Reasons for cooperating processes:
  - Computation speedup
    - Multiple processing cores
    - Distributed computing
- Modularity
  - Subtasks into separate processes or threads
- Client-Server Computing

- Cooperating processes need **Inter-Process Communication** (**IPC**)
- Many IPC mechanisms

IPC

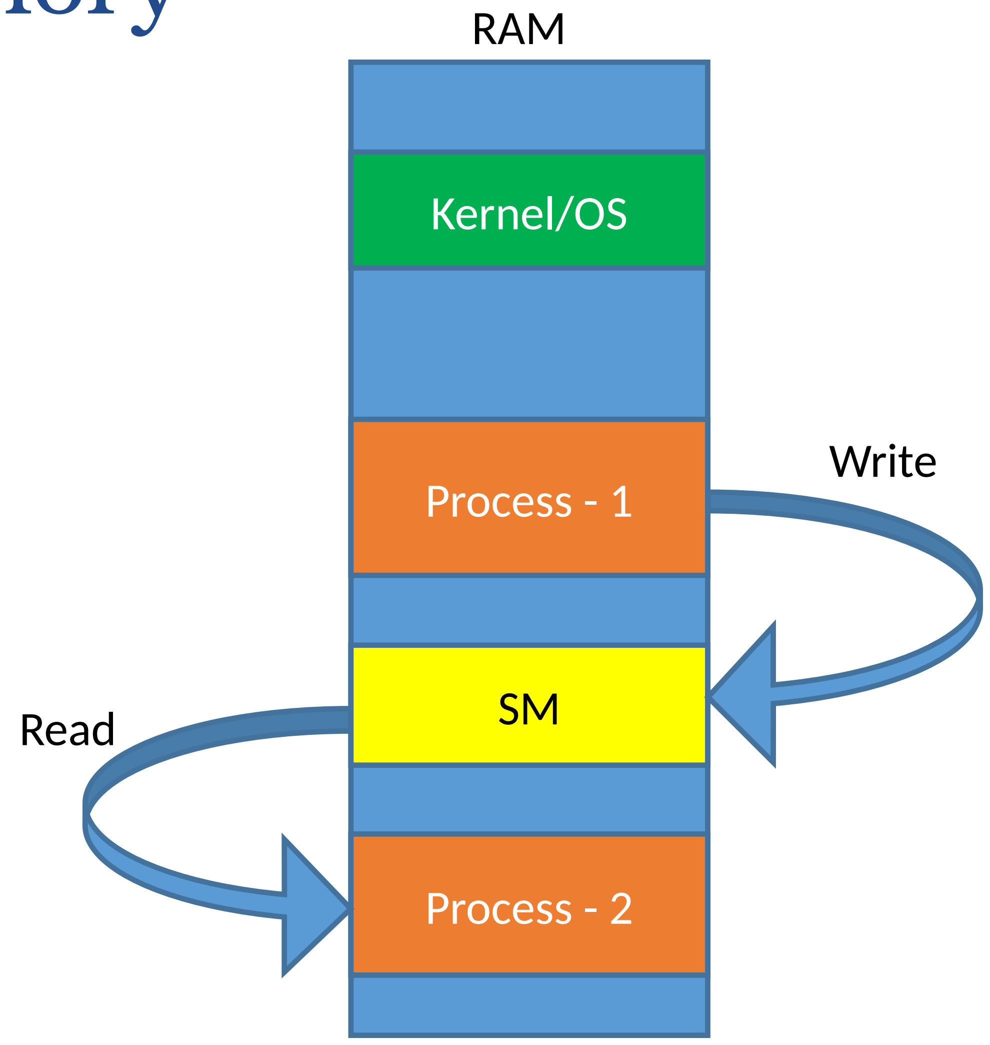
3 Ways

- Shared Memory
- Message Passing
- Pipes
- Signals

Shared Memory

- Process1
  - Create SM
  - Attach SM to it's address Space
  - Write Data into SM

- Process 2
  - Attach SM to it's address Space
  - Read Data from SM written by Writer



Process 1

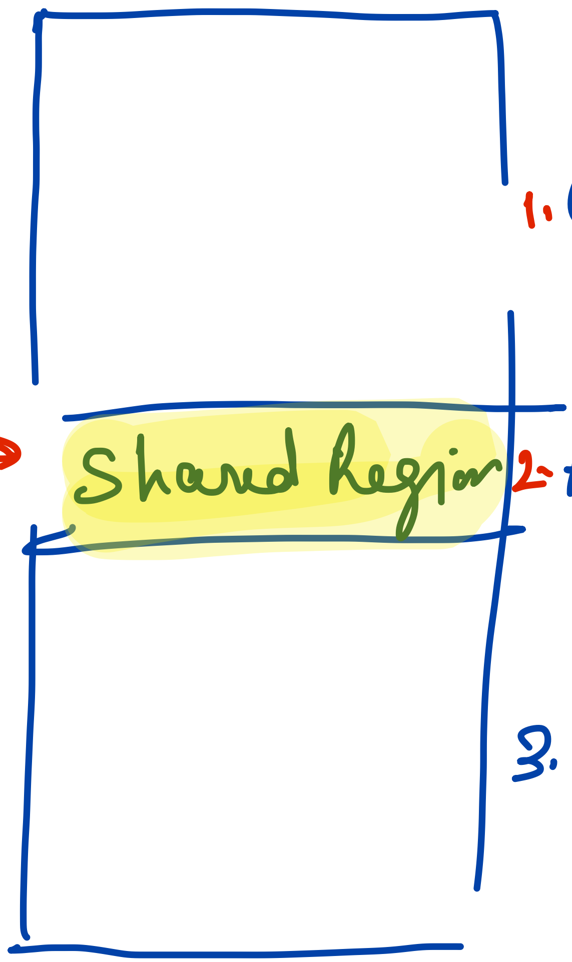
1. Create the shared region and get the i'd

2. Attach with the SR.

Perform operations

3. Detach

4. Remove the pointer // only one process will perform who will complete at last



Process 2

1. Get the i'd of the already exists shared region

2. Attach the shared region

Perform operations

3. Detach

Functions used in SM

- shmget() ▭ to Create the SM
- shmat() ▭ to attach the SM with the address space of the process
- shmdt() ▭ to detach the SM
- shmctl() ▭ to Destroy the SM

shmget()

- **int shmget**(**key_t** key, **size_t** size, **int** shmflg);

- **key**-> Unique value that identifies the SM.
- **size-**> Size of the SM in bytes
- **shmflg** -> Permissions on the SM
- Retuns valid identifier of SM
  - Used in shmat()
- Incase of Unsuccessful -> Returns -1

- **#include<sys/ipc.h>**
- **#include<sys/shm.h>**

shmat()

- **void* shmat**(**int** shmid, **const void *** shmaddr, **int** shmflg);
- **shmid** -> value returned by shmget().
- **shmaddr** -> where to attach the SM in the address space of the calling function
  - Address not know so write NULL. If shmaddr is a NULL pointer, the segment is attached at the first available address as selected by the system.
  - OS will assign it at a suitable location.
- **shmflg** -> if shmaddr is NULL, shmflg is 0.
- Incase of Unsuccessful -> Returns -1
- **#include<sys/types.h>**
- **#include<sys/shm.h>**

shmdt()

- **int shmdt**(**void *** shmaddr)
  - shmdt **detaches** the shared memory segment located at the address specified by shmaddr from the address space of the calling process

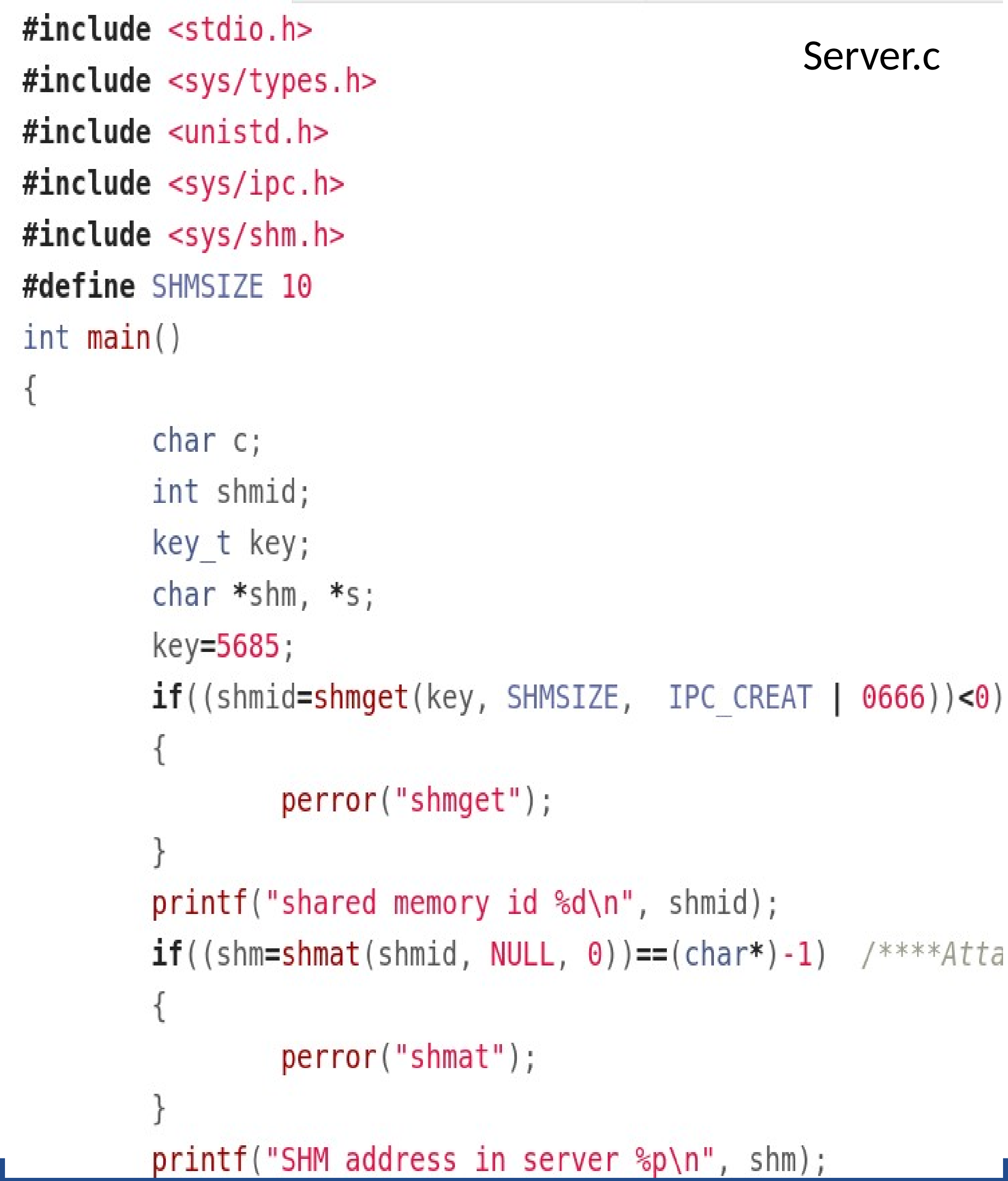
  - On success, it returns 0, on error –1

  - Detaching the shared memory doesn’t delete it
    – it just makes that memory unavailable to the current process

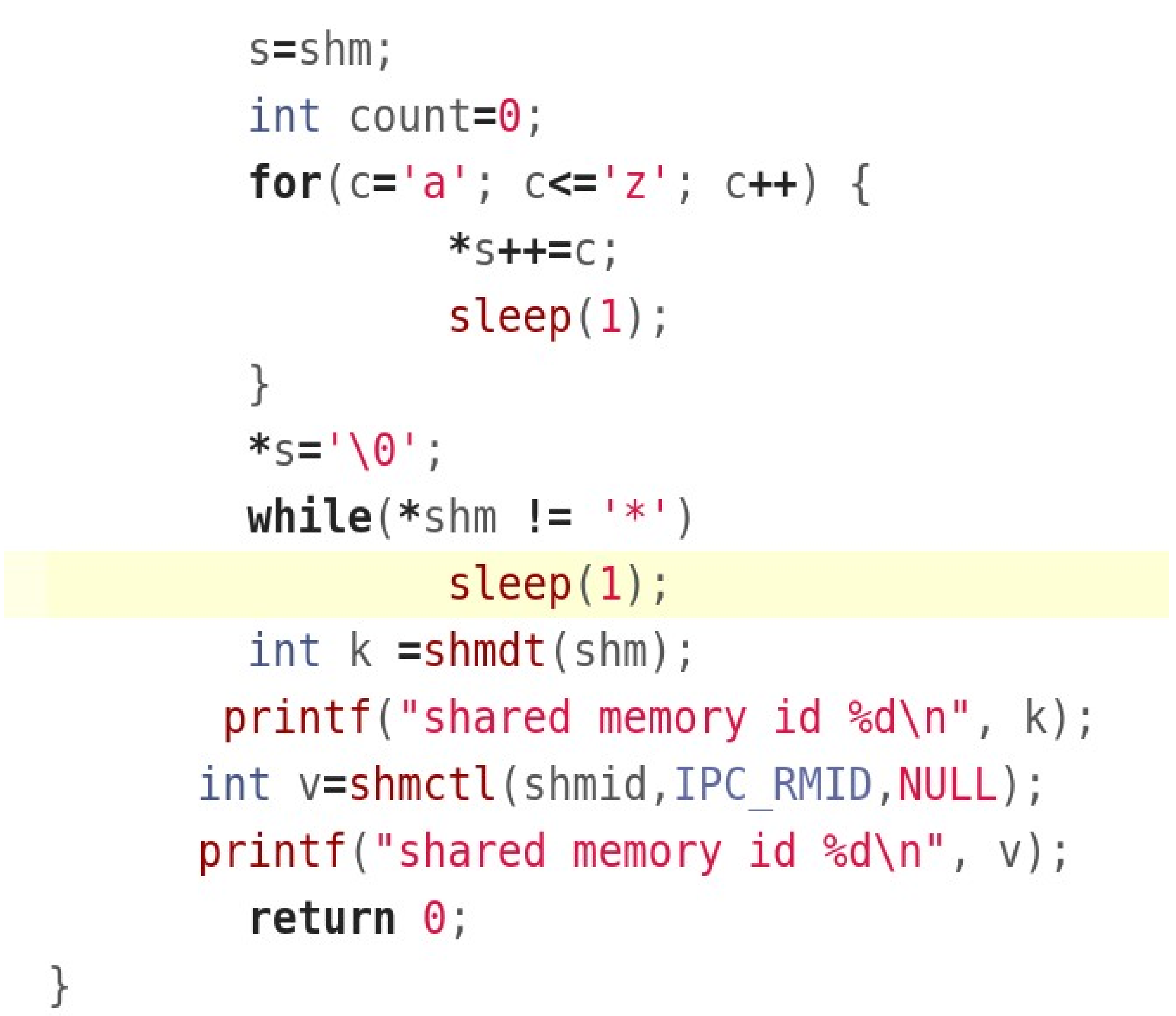
shmctl()

- **int shmctl**(**int** shmid, **int** command, **struct shmid_ds** *buf);
  - returns information about a shared memory segment and can modify it
  - **shmid** -> value returned by shmget().
  - **IPC_STAT**: Retrieve the status of the shared memory segment.
  - **IPC_SET:** Set the status of the shared memory segment.
  - **IPC_RMID**: Remove the shared memory segment.
  - This is a pointer to a **struct shmid_ds** structure that is used to get or set information about the shared memory segment
  - On success, it returns 0, on failure, –1

Server.c

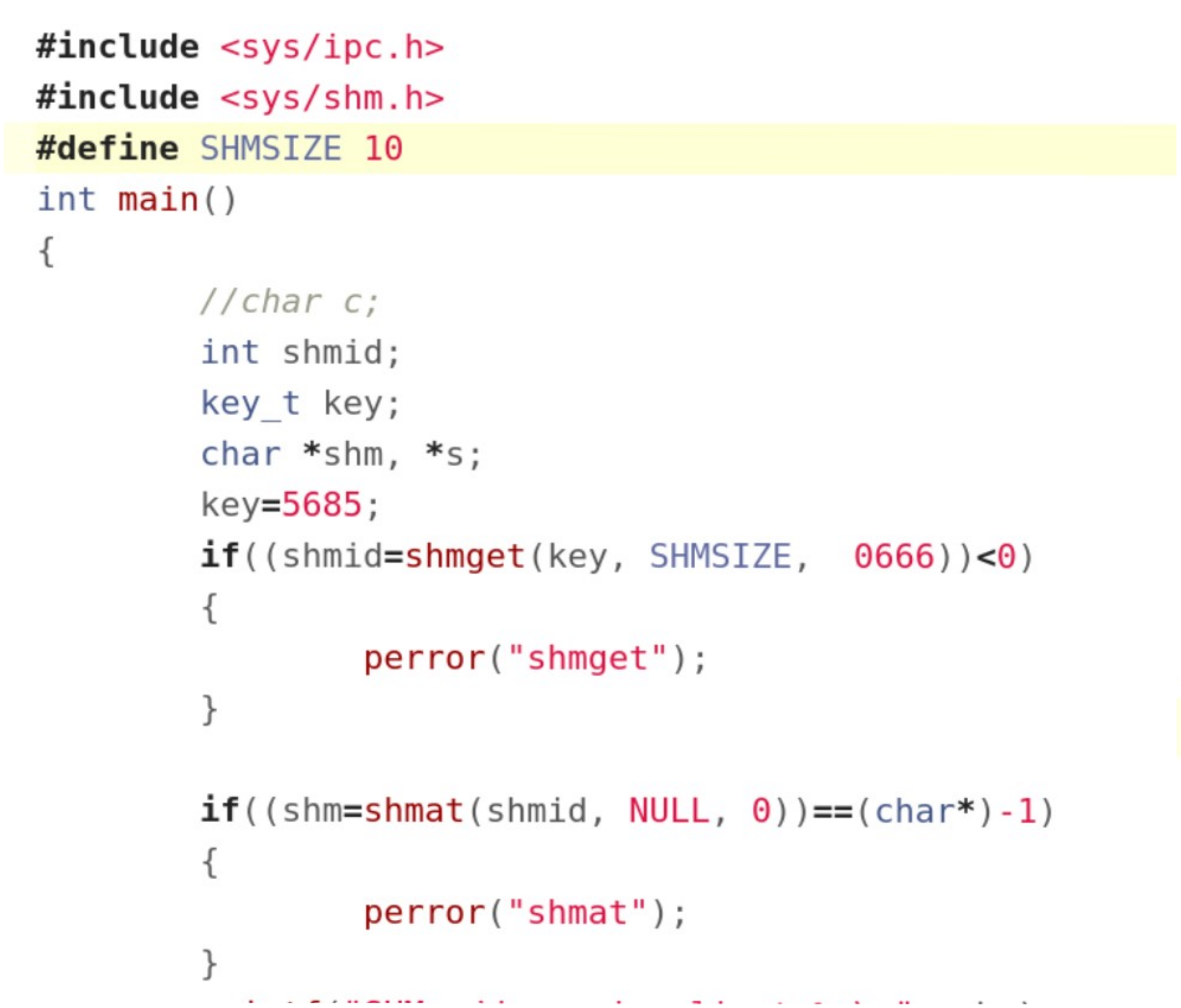


```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define SHMSIZE 10
int main()
{
        char c;
        int shmid;
        key_t key;
        char *shm, *s;
        key=5685;
        if((shmid=shmget(key, SHMSIZE,  IPC_CREAT | 0666))<0)
        {
                perror("shmget");
        }
        printf("shared memory id %d\n", shmid);
        if((shm=shmat(shmid, NULL, 0))==(char*)-1)  /****Atta
        {
                perror("shmat");
        }
        printf("SHM address in server %p\n", shm);
```



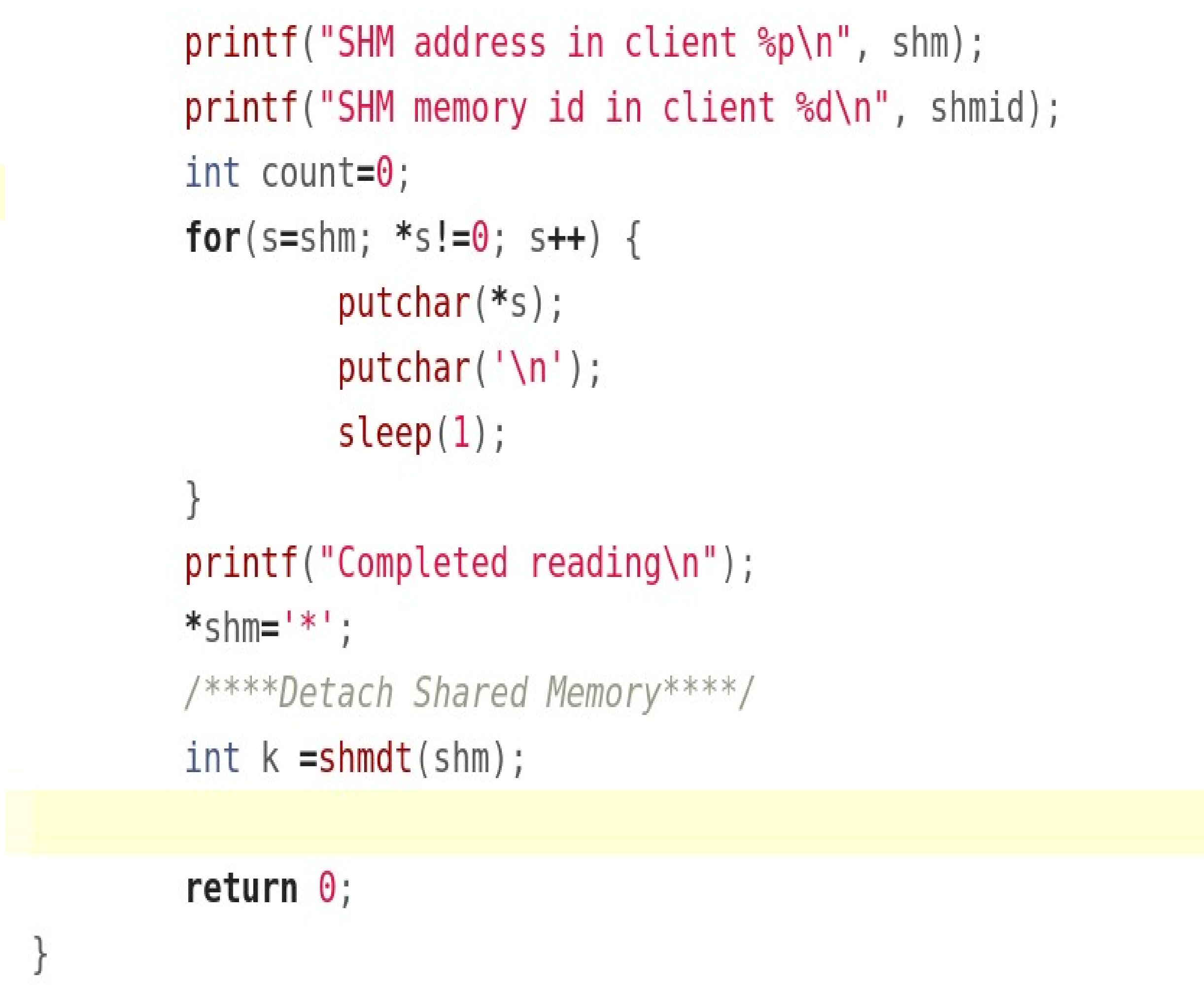
```
        s=shm;
        int count=0;
        for(c='a'; c<='z'; c++) {
                *s++=c;
                sleep(1);
        }
        *s='\0';
        while(*shm != '*')
                sleep(1);
        int k =shmdt(shm);
       printf("shared memory id %d\n", k);
      int v=shmctl(shmid,IPC_RMID,NULL);
      printf("shared memory id %d\n", v);
        return 0;
}
```

client.c



```c
#include <sys/ipc.h>
#include <sys/shm.h>
#define SHMSIZE 10
int main()
{
        //char c;
        int shmid;
        key_t key;
        char *shm, *s;
        key=5685;
        if((shmid=shmget(key, SHMSIZE,  0666))<0)
        {
                perror("shmget");
        }

        if((shm=shmat(shmid, NULL, 0))==(char*)-1)
        {
                perror("shmat");
        }
```



```c
        printf("SHM address in client %p\n", shm);
        printf("SHM memory id in client %d\n", shmid);
        int count=0;
        for(s=shm; *s!=0; s++) {
                putchar(*s);
                putchar('\n');
                sleep(1);
        }
        printf("Completed reading\n");
        *shm='*';
        /****Detach Shared Memory****/
        int k =shmdt(shm);

        return 0;
}
```

Thank You

Any Questions?