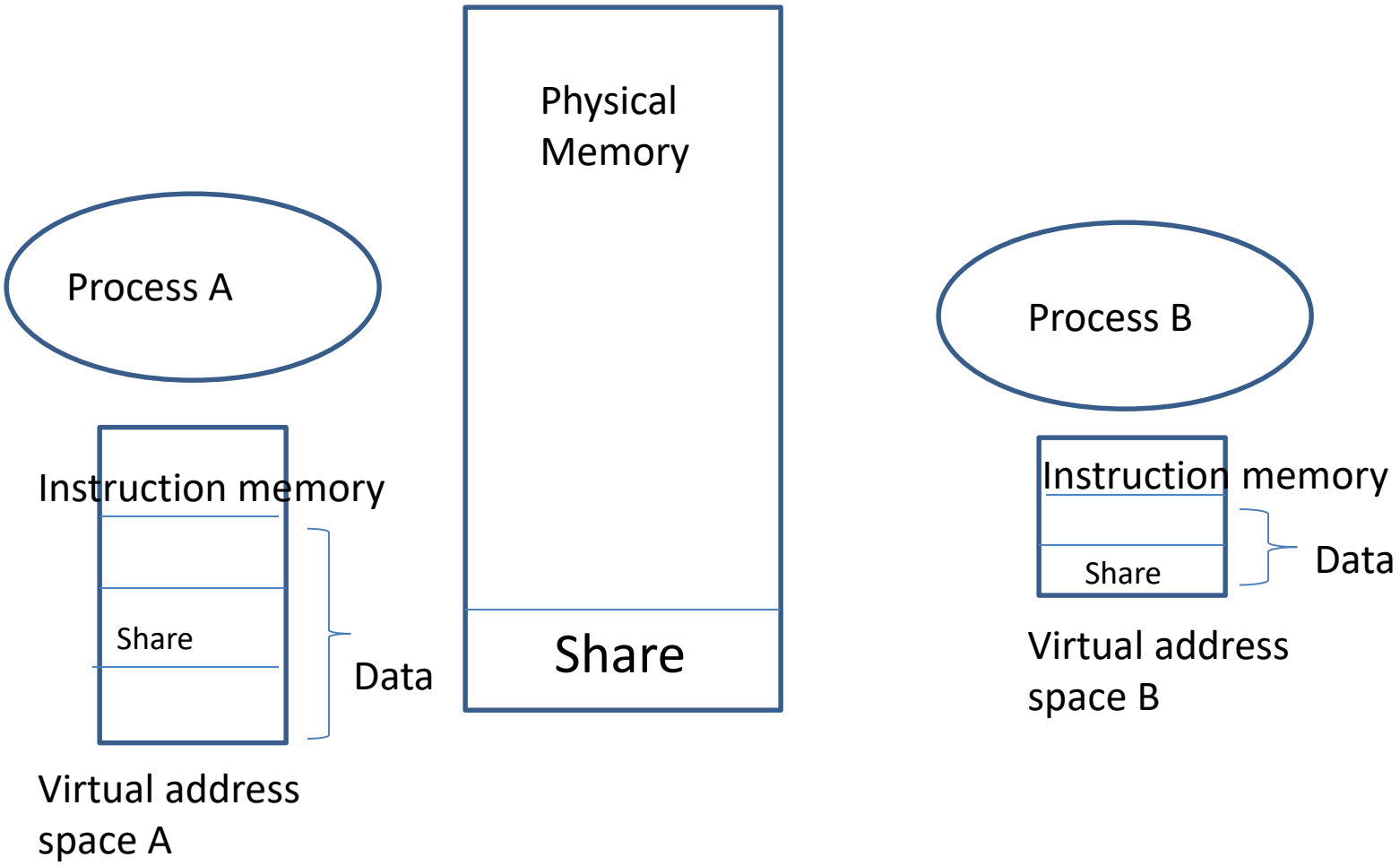
Shared memory

Shared memory

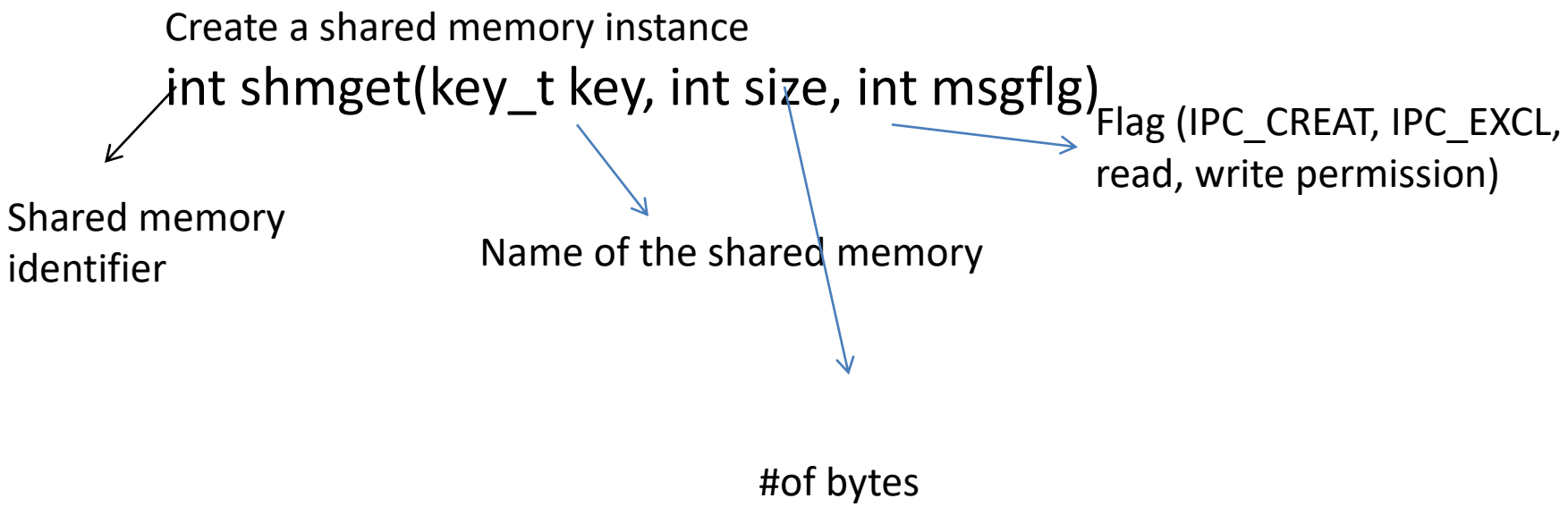


Create a shared memory region

- Create a shared memory region

- Attach a process with the shared memory
- Use
- Detach a process from the shared memory

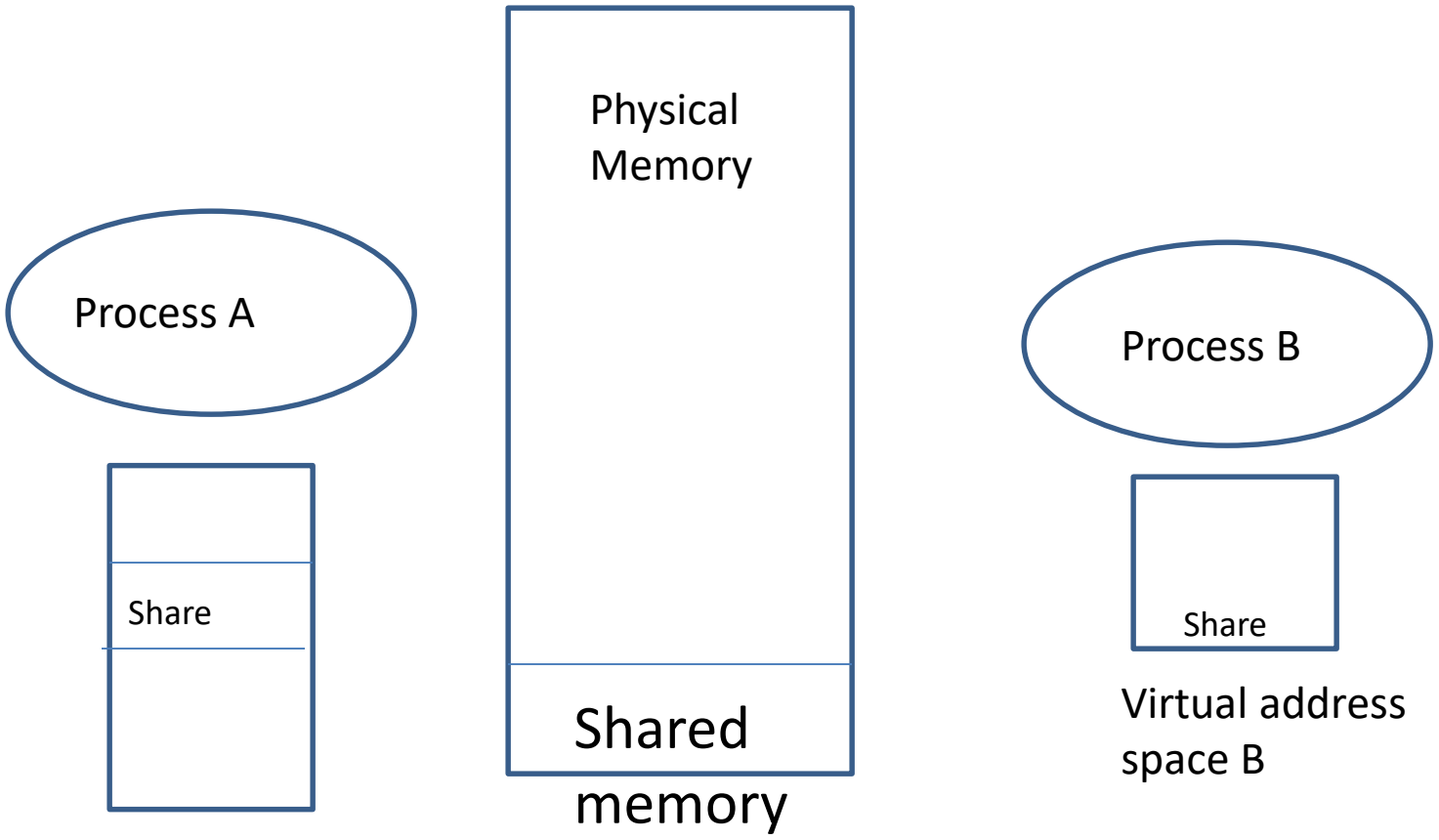
Create a shared memory region



```
int main()
{
        int shmid;
        key_t key;
        key=131;
        shmid=shmget(key,20, IPC_CREAT|0666);
        printf("\nq=%d",shmid);
}
```

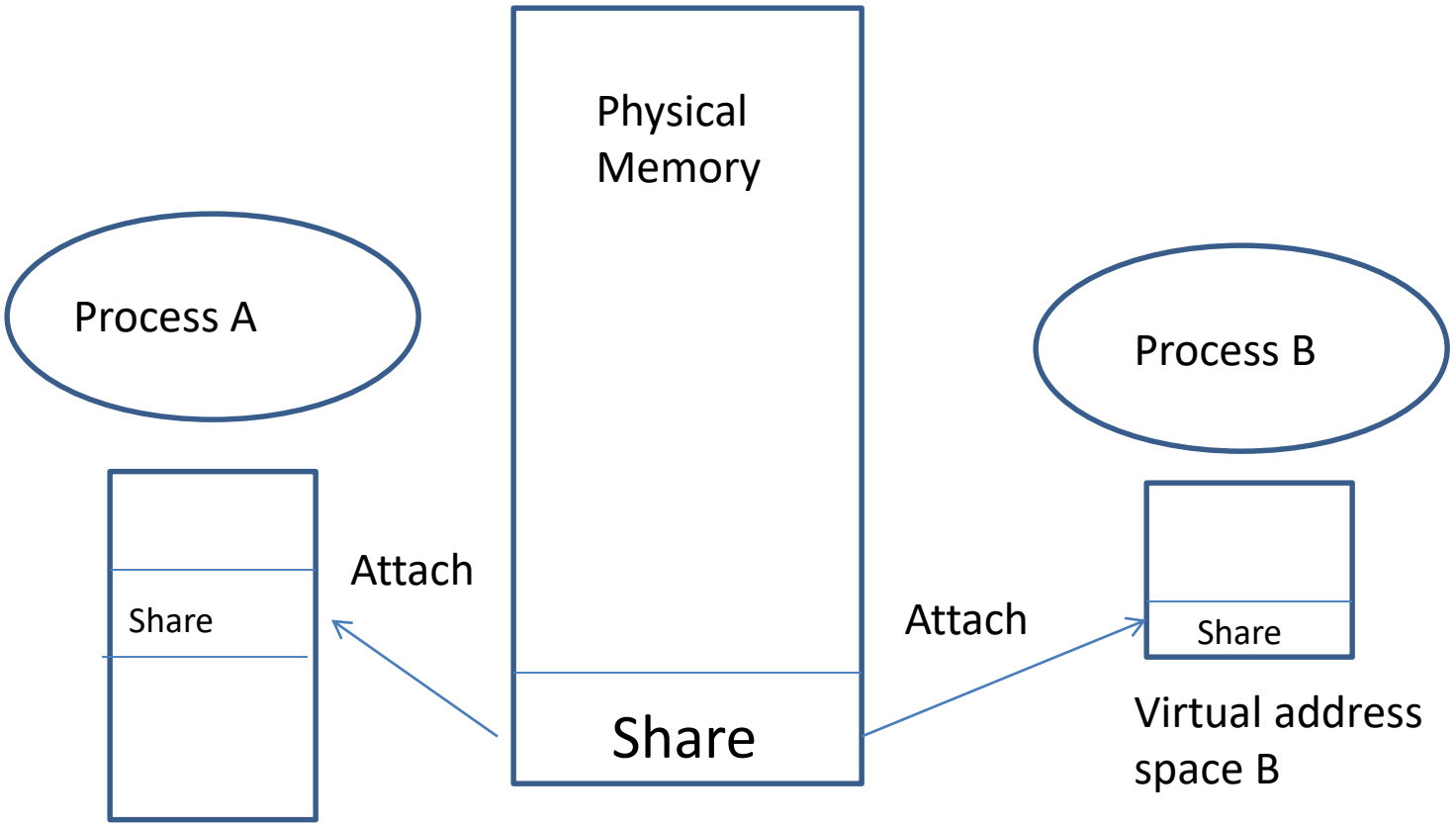
ipcs –m displays the shared memory information in the system

Keys	**ShmID**	**owner**	**permission**	**bytes**	**nattach**


Physical
Memory
Process A
Share
Shared
memory
Process B
Share
Virtual address
space B

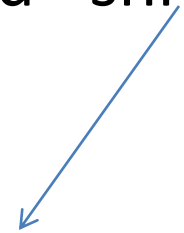
- Create a shared memory region

- Attach a process with the shared memory
- Use
- Detach a process from the shared memory


Physical
Memory
Process A
Process B
Share
Attach
Attach
Share
Share
Virtual address
space B

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

Virtual address of the shared memory segment



attach occurs at the address *shmaddr*

SHM_RDONLY, SHM_RND

shmaddr - (*shmaddr* % SHMLBA))

shmat() attaches the shared memory segment identified by *shmid* to the address space of the calling process.

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
char *ptr
key = 10
size = 20
shmflg =

shmid = shmget (key, size, shmflg));
ptr=(char*)shmat(shmid, NULL, 0);
```

```
int main()
{

	int shmid,f,key=2,i,pid;
	char *ptr;

	shmid=shmget((key_t)key,100,IPC_CREAT|0666);
	ptr=(char*)shmat(shmid,NULL,0);
	printf("shmid=%d ptr=%u\n",shmid, ptr);
	pid=fork();
	if(pid==0)
	{
		strcpy(ptr,“hello\n");
	}
	else
	{
		wait(0);
		printf("%s\n",ptr);
	}

}
```

Child writes “Hello” to the shared memory

Parent reads “Hello” from the shared memory

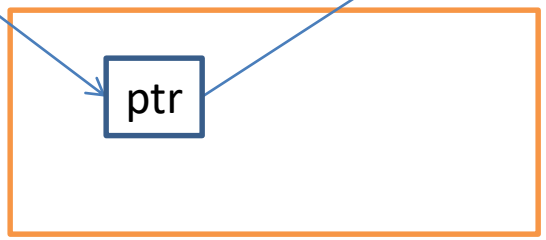
example

writer.c

```
int main()
{

      int shmid,f,key=3,i,pid;
      char *ptr;

      shmid=shmget((key_t)key,100,IPC_CREAT|0666);
      ptr=shmat(shmid,NULL,0);
      printf("shmid=%d ptr=%u\n",shmid, ptr);
      strcpy(ptr,"hello");
     i=shmdt((char*)ptr);

}
```

reader .c

```
int main()
{

      int shmid,f,key=3,i,pid;
      char *ptr;

      shmid=shmget((key_t)key,100,IPC_CREAT|0666);
      ptr=shmat(shmid,NULL,0);
      printf("shmid=%d ptr=%u\n",shmid, ptr);
      printf("\nstr %s\n",ptr);
}
```



Shared memory

```
int main()
{
	struct databuf *ptr;
	int shmid,f,key=2,i,pid;
	char *ptr;

	shmid=shmget((key_t)key,100,IPC_CREAT|0666);
	ptr=(struct databuf*)shmat(shmid,NULL,0);
	printf("shmid=%d ptr=%u\n",shmid, ptr);
	pid=fork();
	if(pid==0)
	{
		ptr->nread=read(0,ptr->buf,10000);
	}
	else
	{
		wait(0);
		printf("parent\n");
		write(1,ptr->buf,10000);
	}
```

```
struct databuf
{
	int nread;
	char buf[1000];
};
```

File descriptor table

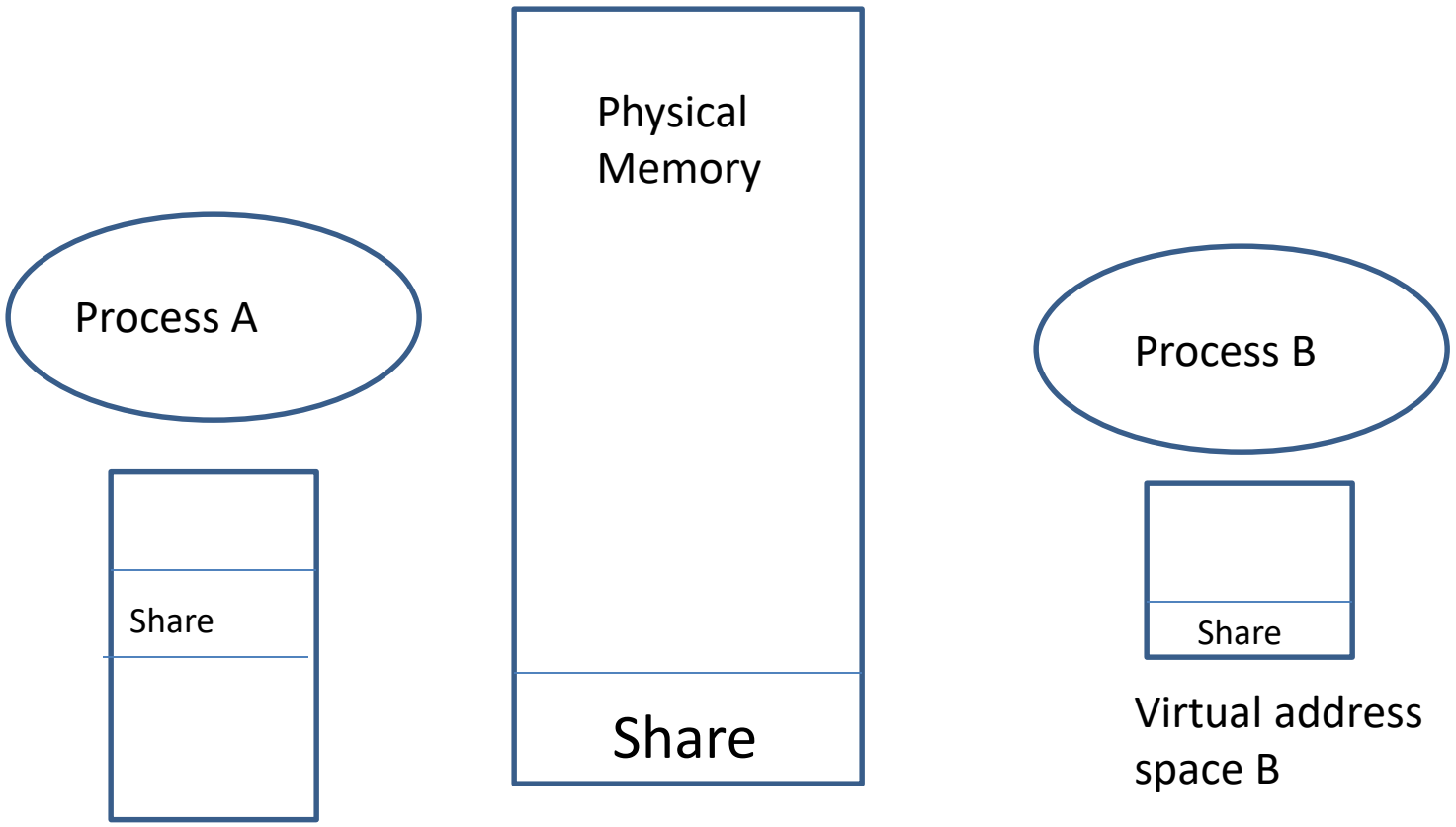
File descriptor (integer)	File name
0	stdin
1	stdout
2	stderr

Use open(), read(), write() system calls to access files

Open() creates a file and returns fd (minimum value)

fd=open(path, O_WRONLY|O_CREAT|O_TRUNC, mode)

Disassociate process from memory



Disassociate process from memory

int shmdt(const void *shmaddr);

```
int main()
{
	shmid=shmget((key_t)key,100,IPC_CREAT|0666);
	ptr=shmat(shmid,NULL,0);
	printf("shmid=%d ptr=%u\n",shmid, ptr);
	strcpy(ptr,"hello");
	printf("\nstr is %s",ptr);

	i=shmdt((char*)ptr);

}
```

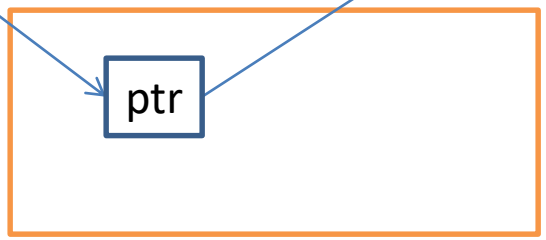
example

writer.c

```
int main()
{

      int shmid,f,key=3,i,pid;
      char *ptr;

      shmid=shmget((key_t)key,100,IPC_CREAT|0666);
      ptr=shmat(shmid,NULL,0);
      printf("shmid=%d ptr=%u\n",shmid, ptr);
      strcpy(ptr,"hello");
     i=shmdt((char*)ptr);

}
```

reader .c

```
int main()
{

      int shmid,f,key=3,i,pid;
      char *ptr;

      shmid=shmget((key_t)key,100,IPC_CREAT|0666);
      ptr=shmat(shmid,NULL,0);
      printf("shmid=%d ptr=%u\n",shmid, ptr);
      printf("\nstr %s\n",ptr);
}
```



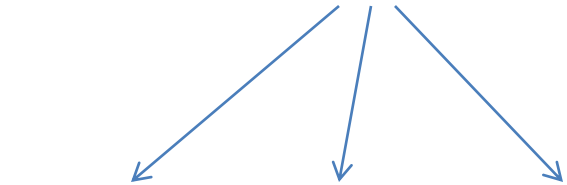
Shared memory

Kernal data structure

```
/* One shmid data structure for each shared memory segment in the system. */

    struct shmid_ds {
        struct ipc_perm shm_perm;       /* operation perms */
        int     shm_segsz;              /* size of segment (bytes) */
        time_t  shm_atime;              /* last attach time */
        time_t  shm_dtime;              /* last detach time */
        time_t  shm_ctime;              /* last change time */
        unsigned short  shm_cpid;       /* pid of creator */
        unsigned short  shm_lpid;       /* pid of last operator */
        short   shm_nattch;             /* no. of current attaches */

                                /* the following are private */

        unsigned short   shm_npages;    /* size of segment (pages) */
        unsigned long   *shm_pages;     /* array of ptrs to frames -> SHMMAX */
        struct vm_area_struct *attaches; /* descriptors for attaches */
    };
```

```
struct ipc_perm {
key_t key;
ushort uid; /* user euid and egid */
ushort gid;
ushort cuid; /* creator euid and egid */
ushort cgid;
ushort mode; /* access modes see mode flags below
*/
 };
```

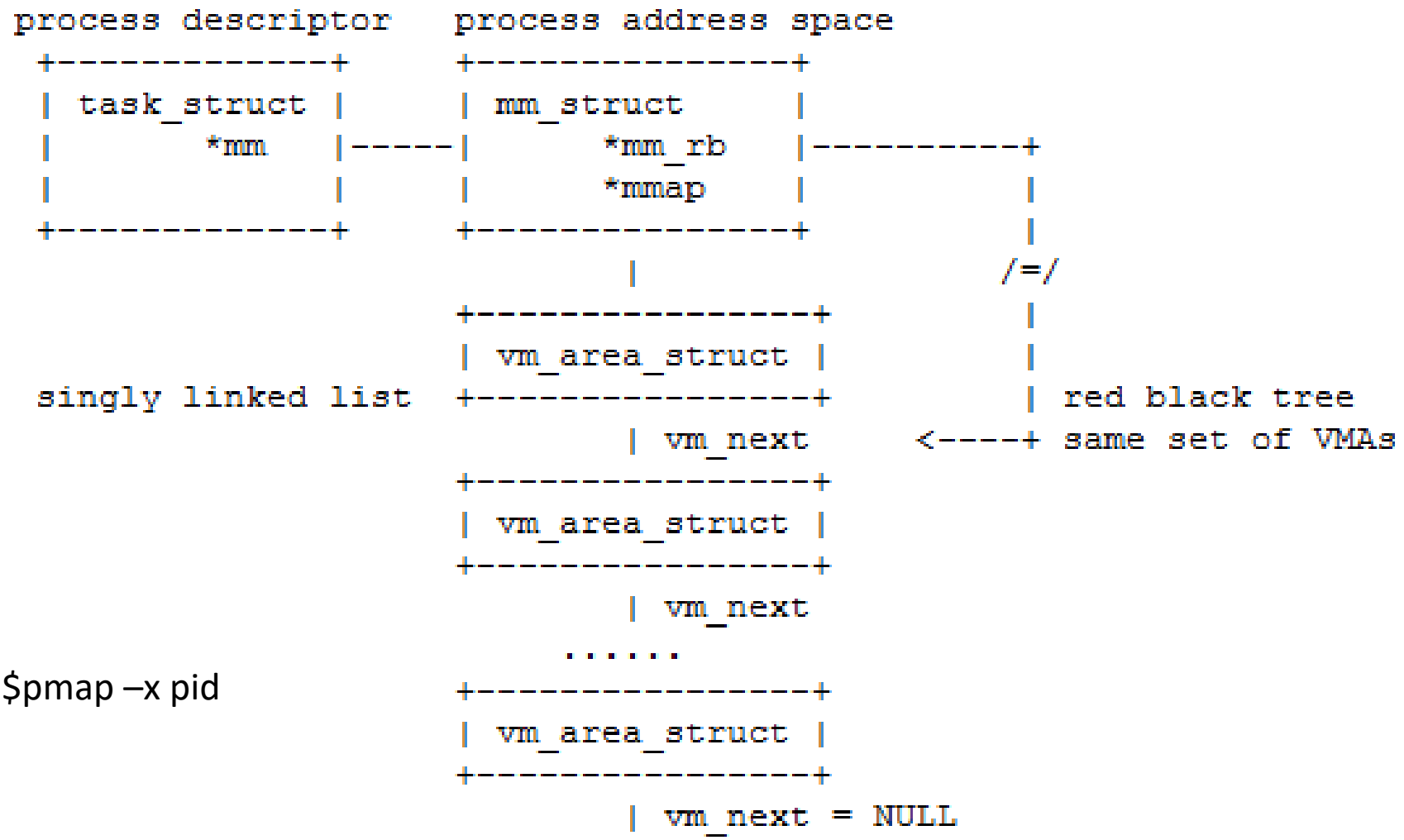
int shmctl(int *shmid*, int *cmd*, struct shmid_ds **buf*);



IPC_STAT IPC_SET IPC_RMID

Remove shared memory

```
struct shmid_ds set;
shmctl(shmid, IPC_STAT, &set);

shmctl(shmid, IPC_SET, &set);

shmctl(shmid, IPC_RMID, 0);
```


process descriptor
process address space
task_struct
*mm
mm_struct
*mm_rb
*mmap
vm_area_struct
vm_next
vm_area_struct
vm_next
.....
vm_area_struct
vm_next = NULL
singly linked list
red black tree
same set of VMAs

$pmap –x pid

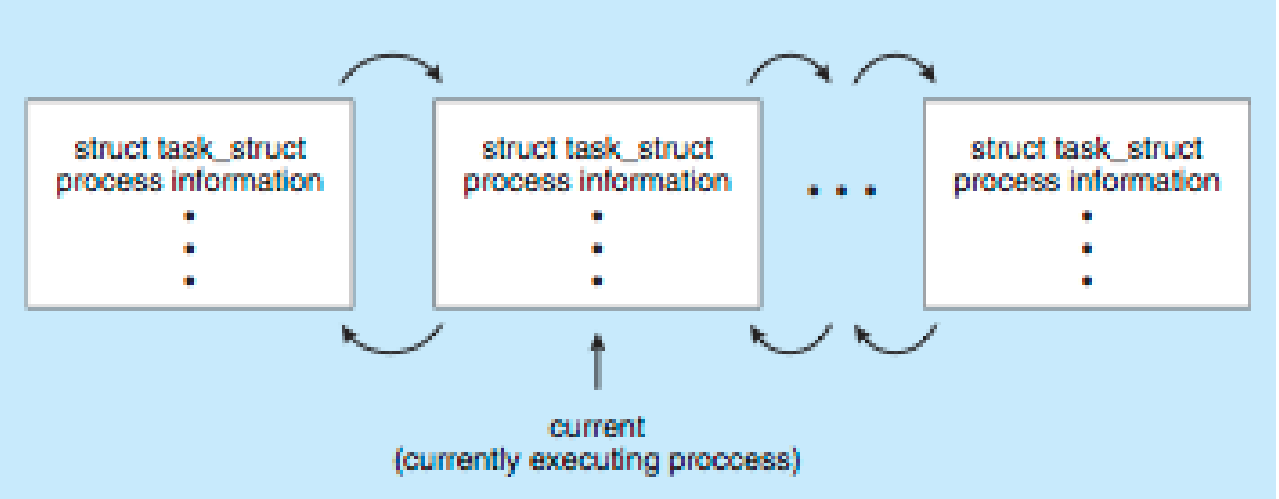
```
                  pmap -x 12519
12519:   ./a.out
Address           Kbytes     RSS   Dirty Mode   Mapping
0000000000400000       4       4       4 r-x--  a.out
0000000000600000       4       4       4 rw---  a.out
0000003ffc800000     128     104       0 r-x--  ld-2.12.so
0000003ffca1f000       4       4       4 r----  ld-2.12.so
0000003ffca20000       4       4       4 rw---  ld-2.12.so
0000003ffca21000       4       4       4 rw---    [ anon ]
0000003ffcc00000    1572     160       0 r-x--  libc-2.12.so
0000003ffcd89000    2048       0       0 -----  libc-2.12.so
0000003ffcf89000      16       8       8 r----  libc-2.12.so
0000003ffcf8d000       4       4       4 rw---  libc-2.12.so
0000003ffcf8e000      20      12      12 rw---    [ anon ]
0000003ffd800000     524      20       0 r-x--  libm-2.12.so
0000003ffd883000    2044       0       0 -----  libm-2.12.so
0000003ffda82000       4       4       4 r----  libm-2.12.so
0000003ffda83000       4       4       4 rw---  libm-2.12.so
00007f6aaccea000      12      12      12 rw---    [ anon ]
00007f6aacd14000       4       4       4 rw---    [ anon ]
00007fff13337000      84       8       8 rw---    [ stack ]
00007fff133bb000       4       4       0 r-x--    [ anon ]
ffffffffff600000       4       0       0 r-x--    [ anon ]
----------------  ------  ------  ------
total kB            6492     364      76
```

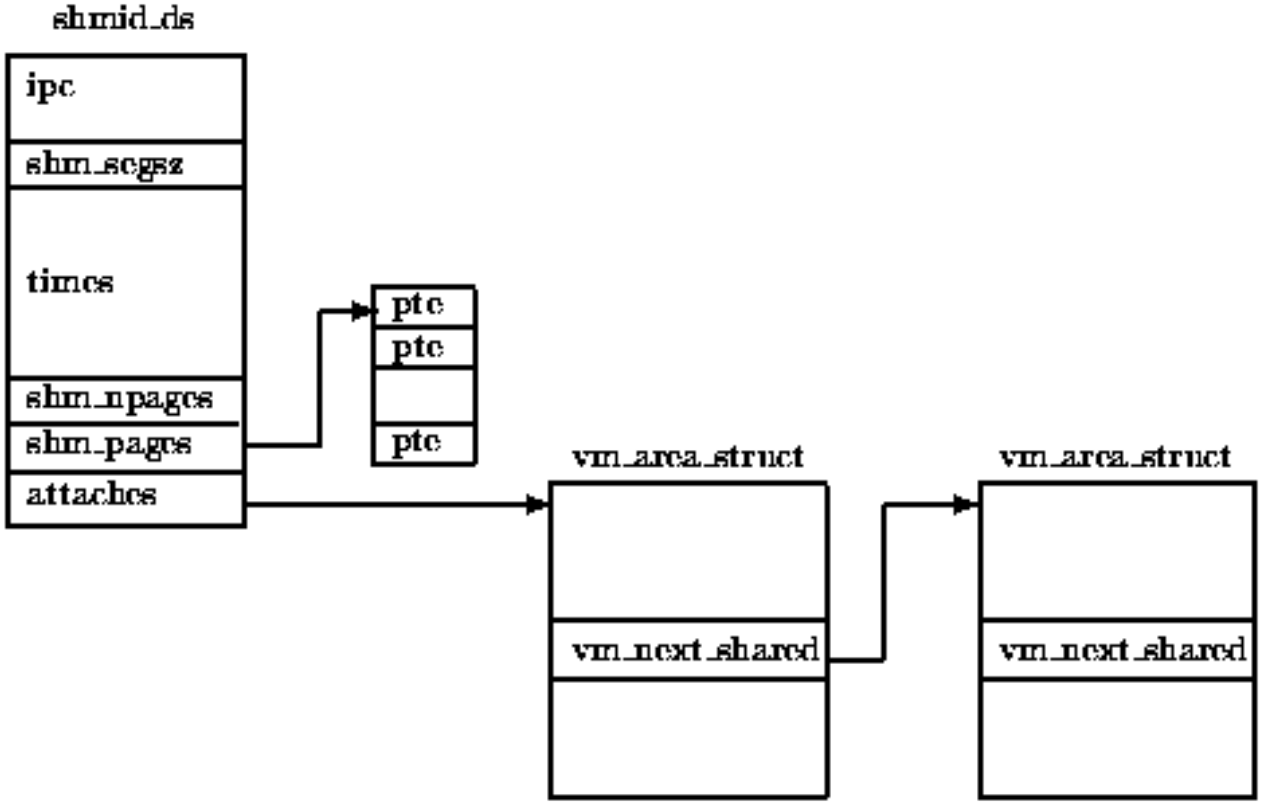
Process Representation in Linux

Represented by the C structure `task_struct`

```
pid t pid; /* process identifier */
long state; /* state of the process */
unsigned int time slice /* scheduling information */
struct task struct *parent; /* this process's parent */
struct list head children; /* this process's children */
struct files struct *files; /* list of open files */
struct mm_struct *mm; /* address space of this pro */
```

Doubly linked list




shmid_ds
ipc
shm_segsz
times
shm_npages
shm_pages
attaches
pte
pte
pte
vm_area_struct
vm_next_shared
vm_area_struct
vm_next_shared

```
struct vm_area_struct {
        struct mm_struct    *vm_mm;     /* associated mm_struct */
        unsigned long       vm_start;   /* VMA start, inclusive */
        unsigned long       vm_end;     /* VMA end, exclusive */
        unsigned long       vm_flags;
        struct vm_area_struct *vm_next; /* points to next VMA */
        struct vm_operations_struct *vm_ops;  /* associated ops */
        ...

    }
```