Program for IPC using shared memory

Shared Memory is the fastest inter-process communication (IPC) method. The operating system maps a memory segment in the address space of several processes so that those processes can read and write in that memory segment.
The overview is as shown below:
Two functions:shmget() and shmat() are used for IPC using shared memory. shmget() function is used to create the shared memory segment while shmat() function is used to attach the shared segment with the address space of the process.

Syntax (shmget()):

```
#include <sys/ipc.h>
#include <sys/shm.h>
int shmget(key_t key, size_t size, int shmflg);
```

The first parameter specifies the unique number (called key) identifying the shared segment. The second parameter is the size of the shared segment e.g. 1024 bytes or 2048 bytes. The third parameter specifies the permissions on the shared segment. On success the shmget() function returns a valid identifier while on failure it return -1.

Syntax (shmat()):

```
#include <sys/types.h>
#include <sys/shm.h>
void *shmat(int shmid, const void *shmaddr, int
shmflg);
```

shmat() is used to attach the created shared segment with the address space of the calling process. The first parameter here is the identifier which shmget() function returns on success. The second parameter is the address where to attach it to the calling process. A NULL value of second parameter means that the system will automatically choose a suitable address. The

third parameter is '0' if the second parameter is NULL, otherwise, the value is specified by SHM_RND.
We will write two program for IPC using shared memory. *Program 1* will create the shared segment, attach to it and then write some content into it. Then *Program 2* will attach itself to the shared segment and read the value written by Program 1.

//**Program 1**: This program creates a shared memory segment, attaches itself to it and then writes some content into the shared memory segment.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main()
{
int i;
void *shared_memory;
char buff[100];
int shmid;
shmid=shmget((key_t)2345, 1024, 0666|IPC_CREAT); //
creates shared memory segment with key 2345, having
size 1024 bytes. IPC_CREAT is used to create the
shared segment if it does not exist. 0666 are the
permisions on the shared segment
printf("Key of shared memory is %d\n",shmid);
shared_memory=shmat(shmid,NULL,0); //process attached
to shared memory segment
printf("Process attached at %p\n",shared_memory); //
this prints the address where the segment is attached
with this process
printf("Enter some data to write to shared
memory\n");
read(0,buff,100); //get some input from user
strcpy(shared_memory,buff); //data written to shared
memory
```

```
printf("You wrote : %s\n",(char *)shared_memory);
}
```

How it works?

shmget() function creates a segment with key 2345, size 1024 bytes and read and write permissions for all users. It returns the identifier of the segment which gets store in shmid. This identifier is used in shmat() to attach the shared segment to the address space of the process. NULL in shmat() means that the OS will itself attach the shared segment at a suitable address of this process.
Then some data is read from the user using read() system call and it is finally written to the shared segment using strcpy() function.

//Program 2: This program attaches itself to the shared memory segment created in Program 1. Finally, it reads the content of the shared memory

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main()
{
int i;
void *shared_memory;
char buff[100];
int shmid;
shmid=shmget((key_t)2345, 1024, 0666);
printf("Key of shared memory is %d\n",shmid);
shared_memory=shmat(shmid,NULL,0); //process attached
to shared memory segment
printf("Process attached at %p\n",shared_memory);
printf("Data read from shared memory is : %s\n",(char
*)shared_memory);
}
```

How it works?

shmget() here generates the identifier of the same segment as created in Program 1. Remember to give the same key value. The only change is, do not write IPC_CREAT as the shared memory segment is already created. Next, shmat() attaches the shared segment to the current process.
After that, the data is printed from the shared segment. In the output, you will see that it is the same data that you have written while executing the Program 1.