# shared memory

Each process has an address space which corresponds to a page table. There are as many page tables as there are processes.
Shared memory corresponds to putting the same “real memory pages” in the page tables of two different processes.

# functions - System V

```
int shmget(key_t key, int size, int flags)
// obtains an identifier to an existing shared memory or creates a new one
```

```
- key: IPC_PRIVATE (creates a new unique identifier) or an existing identifier (ftok() can be used to generate a number based on a filename)
- size: shared memory size in bytes (it will be rounded to a multiple of page size)
- flags: IPC_CREAT (creates a new segment - if not used, shmget() will find the segment associated with the key and check the access permissions) and IPC_EXCL (used with IPC_CREAT to ensure that the call creates de segment - if it already exists, the call fails)
```

```
int * shmat(int shmid, const void *where, int flags)
// maps a certain shared memory region into the current process address space
```

```
- shmid: value returned by shmget()
- where: unused address space location where to map the shared memory (usually, NULL)
- flags: represents different ways to do the mapping (usually, 0)
```

```
int shmdt(const void *where)
// unsmaps a certain shared memory region from the current address space
```

```
- where: address space location where the sahred memory was mapped
```

```c
int shmctl(int shmid, int cmd, struct shmid_ds *buff)
// provides a variety of control operations on the shared memory
```

- shmid: value returned by shmget()
- cmd: command (usually IPC_RMID to remove shared memory)
- buff: structure used in some control operations