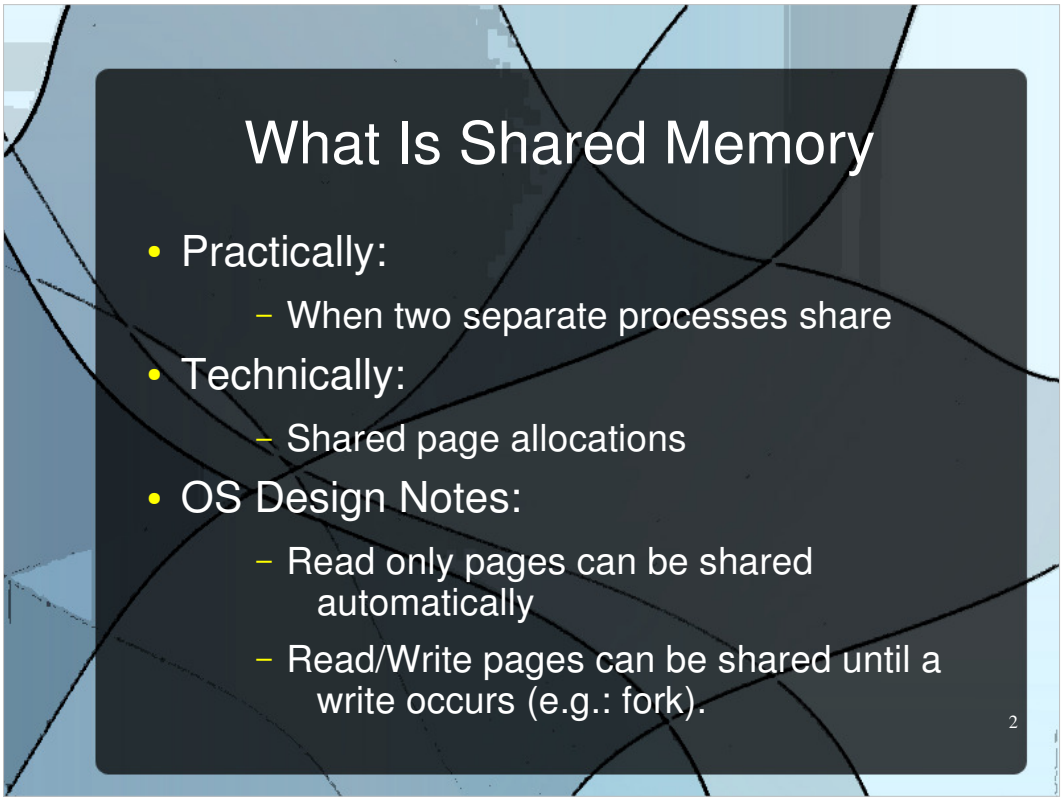
Shared Memory

by Nathan Shearer

- What Is Shared Memory?
- Some Uses
- Sharing Memory the SysV-IPC Way
  - Creating
  - Attaching
  - Clean up your mess!
- Sharing Means Cooperating
- Alternatives?



Practically:

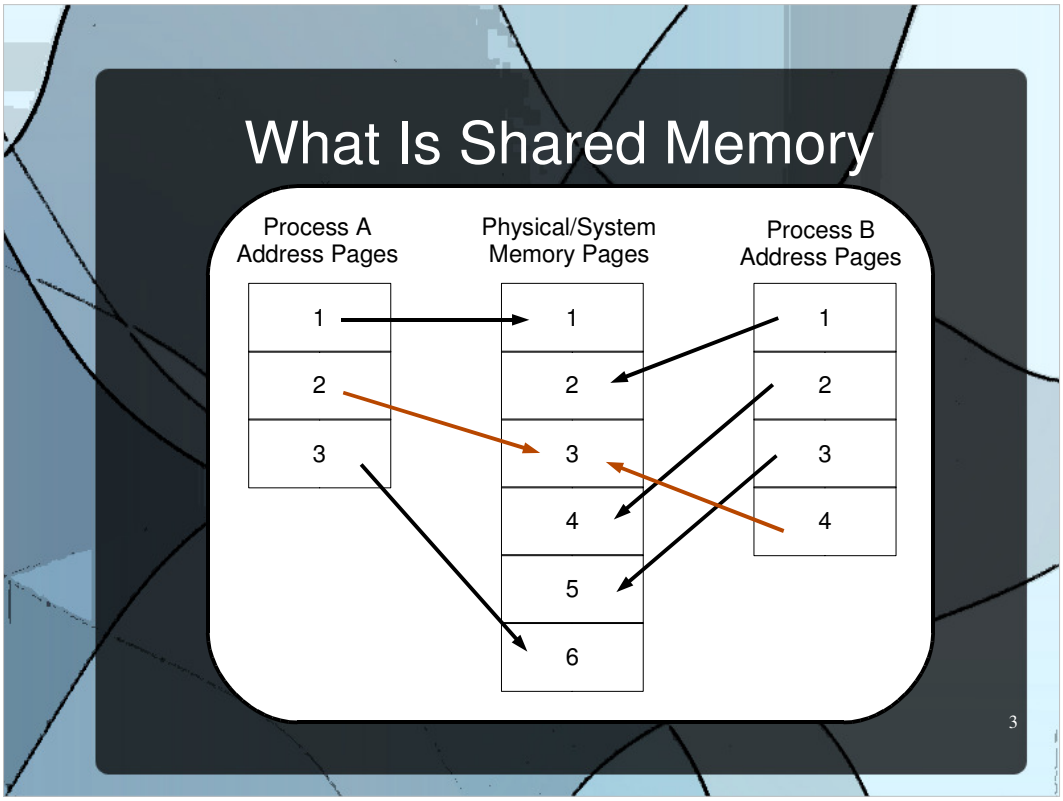
- Shared memory is when two or more processes can read and write to the same memory block.
- This would result in a segmentation fault with non-shared memory.

Technically

- Shared memory is implemented as pages in separate process address spaces which are mapped to a single page in the system memory address space.

OS Design Notes

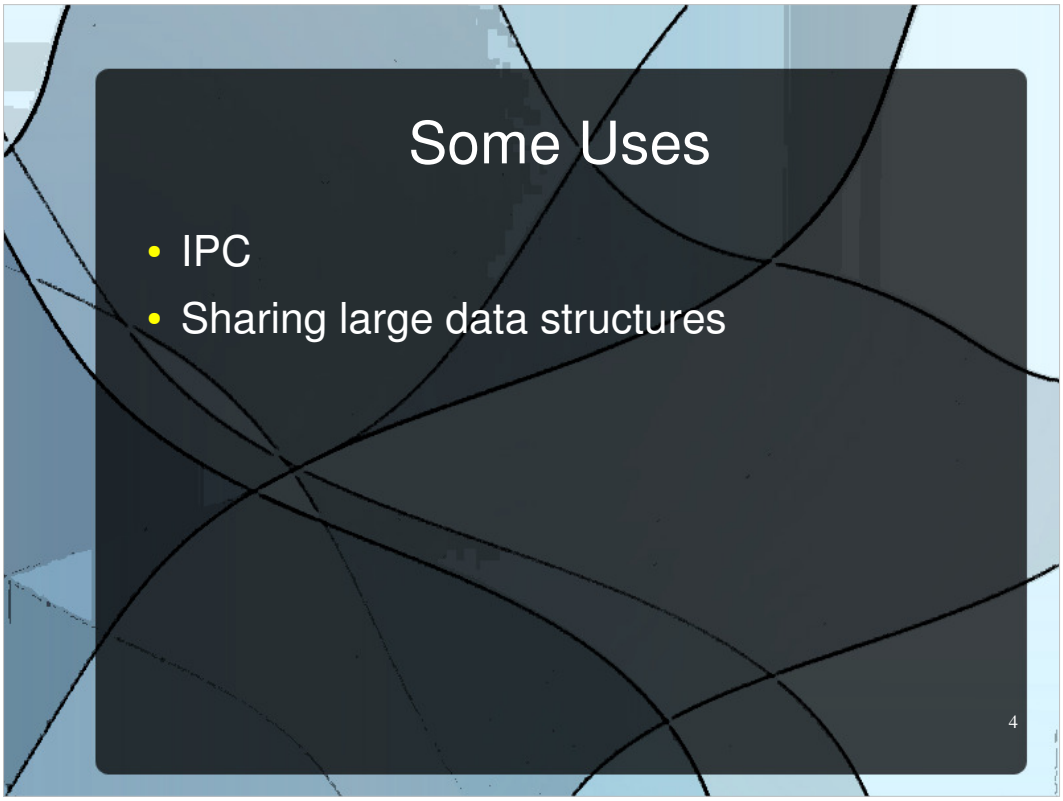
- The OS may share pages automatically for performance (Linux, plus probably others)
- Read only Instruction pages (e.g.: running apache twice). Not data pages.
- On fork, data pages are shared until the new process makes a change. This is what makes fork so fast. Method known as ”copy on write.”



- Processes A and B are sharing physical memory page 3

- Arrows represent virtual memory mappings done by OS
- Note that shared pages in process are at different locations in each process address space.
  Don't share memory addresses (pointers) between processes.

- Note OS needs to know not to evict page 3 if a process A terminates, but to evict it if both A and B terminate.
  - But what if A is going to start again real soon? That would cause excessive paging.

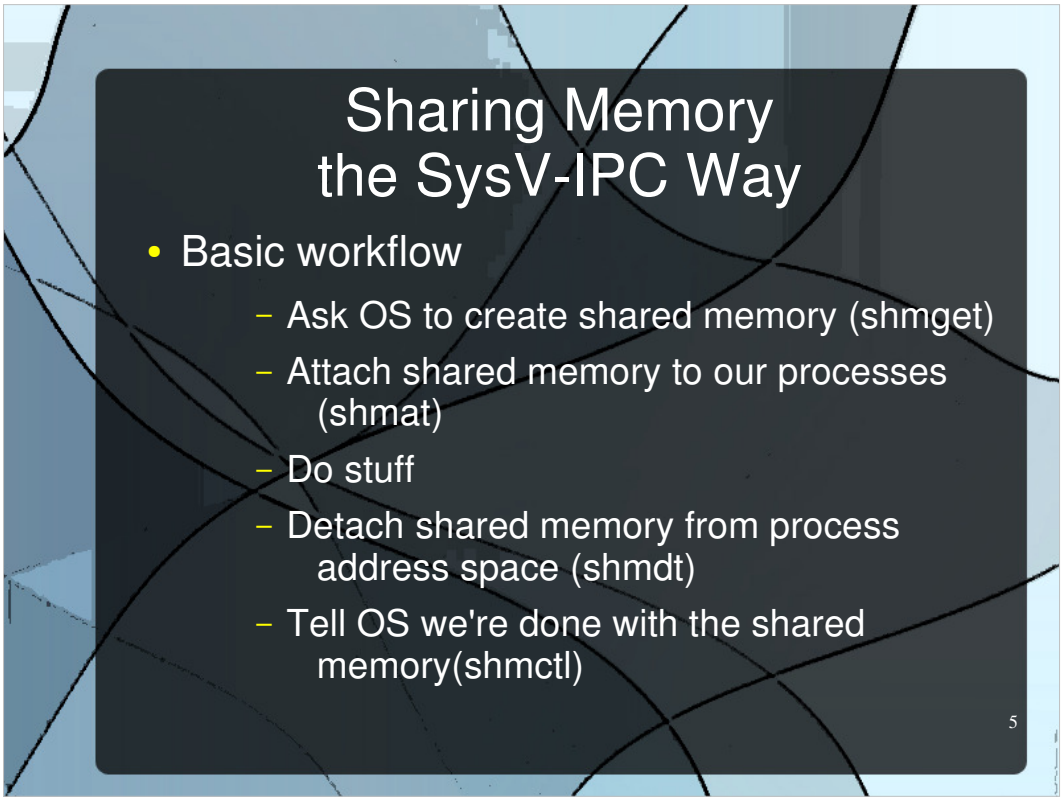


Common uses:

- Inter-process communication (IPC)
  - X-Windows copies bitmaps?

- Used to simulate common external (HW) inputs in real time systems class.

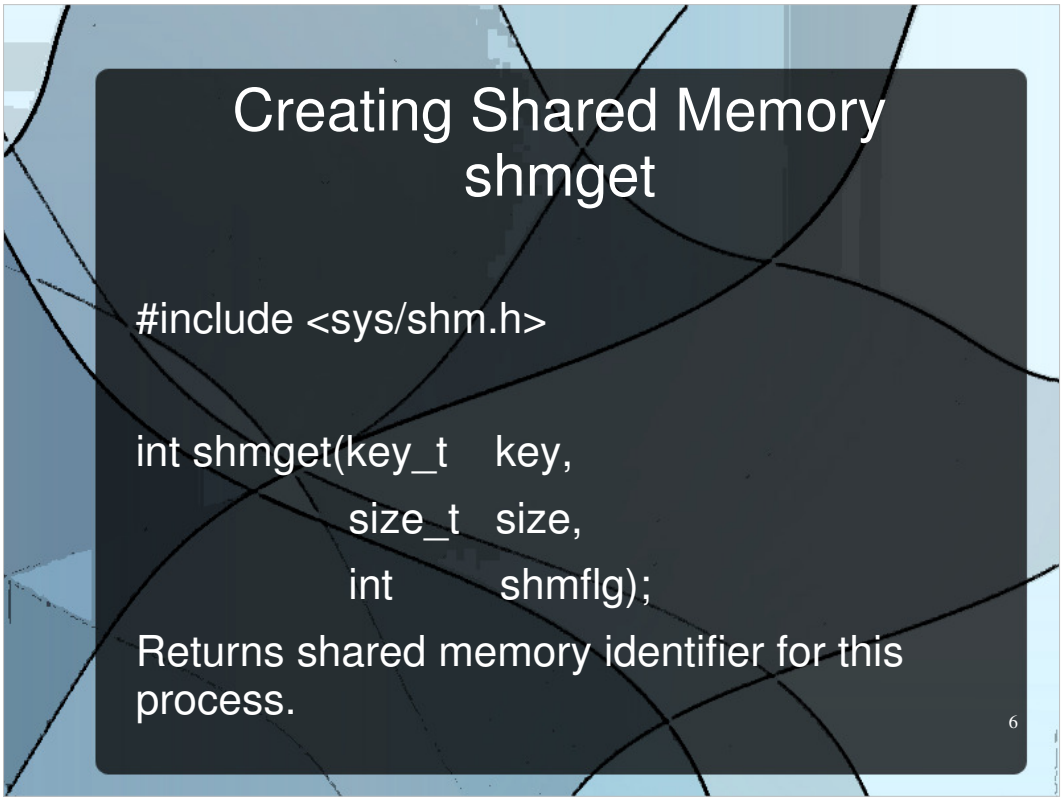
- For efficiency, multiple process can create a single copy of a large amount of data.



Unix System 5 (SysV) is a standard Unix was built on.

There are some POSIX standards as well. Not sure how closely related.

Windows developers, go buy an expensive book.



shmget() requests size bytes of shared memory.

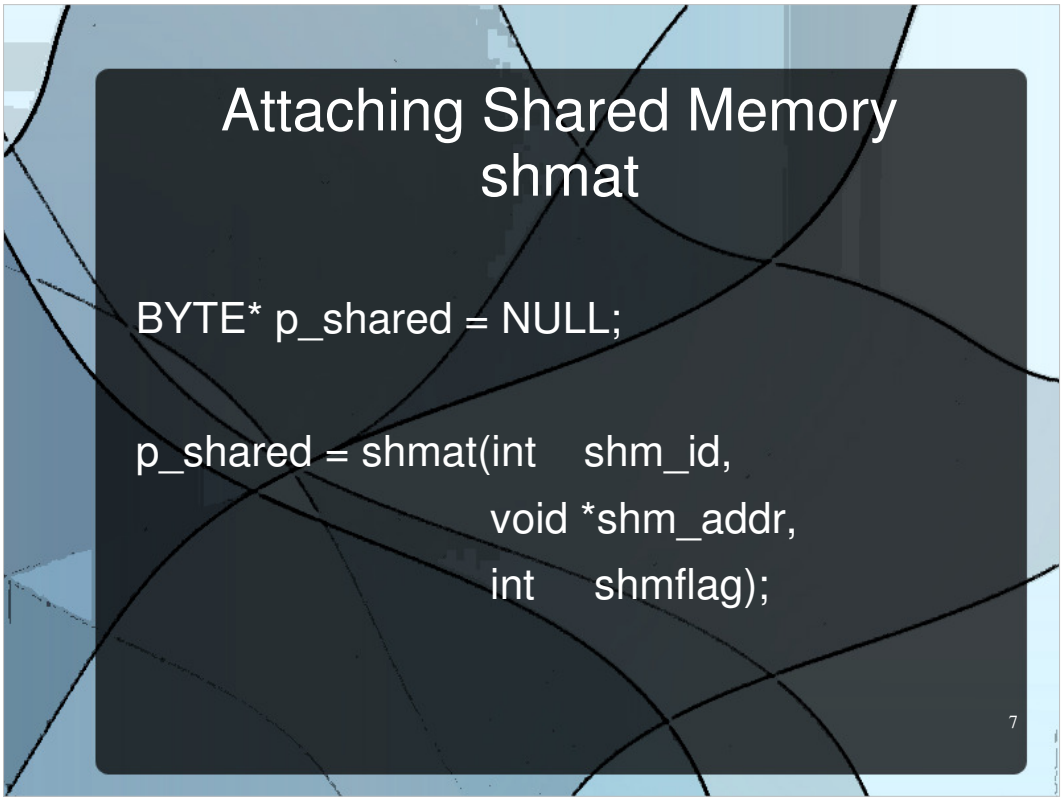
- key: A unique identifier (integer) for this shared memory segment. This is how the OS names shared memory segments, and therefore how multiple process attach to the same shared memory location.
- size: How many bytes do you want?
- shmflag: Various ”ORed” flags
  - IPC_CREAT: Create the shared memory if it doesn't exist.
  - IPC_EXCL: Fail if key already exists.
  - Octal permissions code

Returns an ID used to reference this shared memory.

shmget() is also called by processes not creating the memory in order to get the ID.

Example:

```
shm_id = shmget(shm_key, sizeof(int) + sizeof(char), 0666 |
   IPC_CREAT | IPC_EXCL))
```



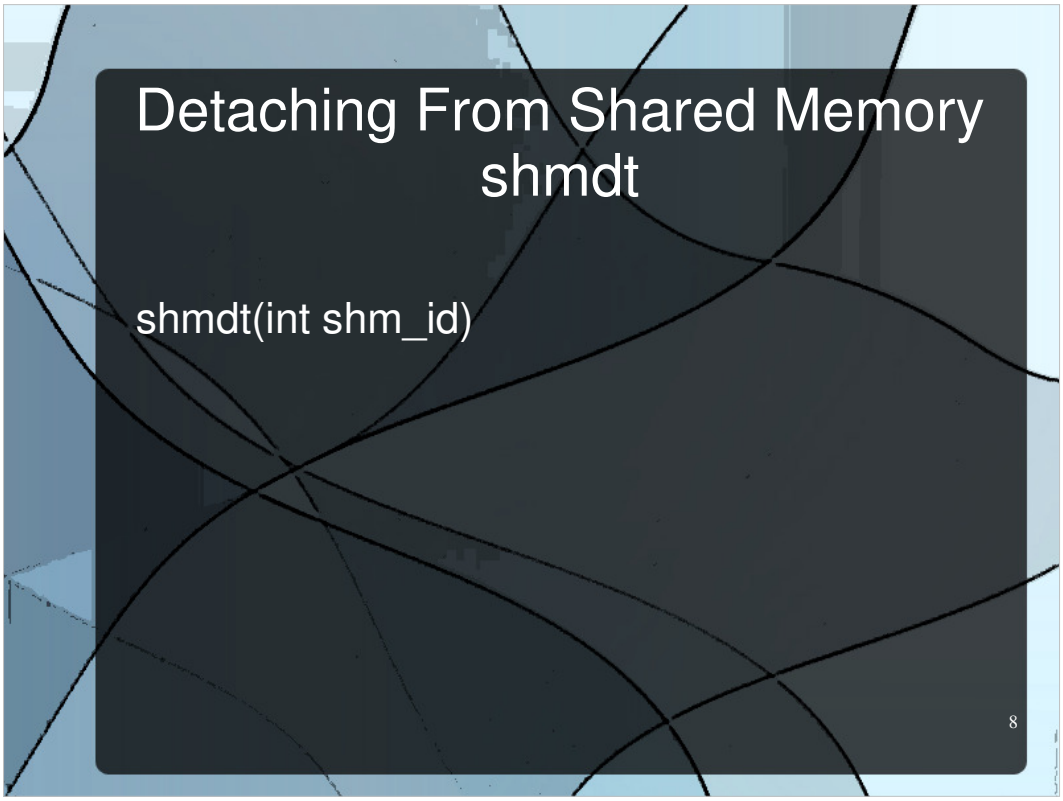
shmat() Attaches shared memory into process virtual address space.

- shm_id: What we got from shmget(), It is the ID localized to this process for the shared memory location.
- shm_addr: Suggestion for where to place shared memory. Not recommended
- shmflag: various options. For example, flag exists to attach read only.

Returns pointer to your shared memory block

Example:

```
p_shared = (BYTE*) shmat(shm_id, (void*)0, 0);
```



shmdt() tells the OS that this process is done using the shared memory.

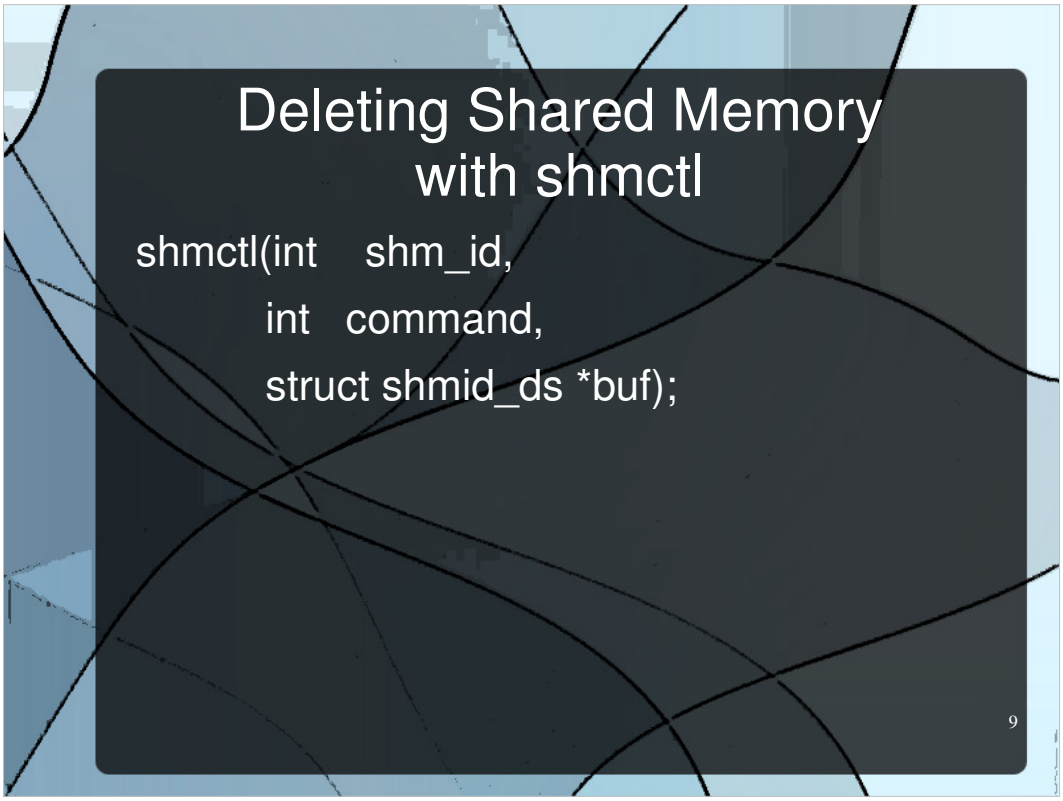
It does not delete the shared memory page (other processes can still attach).

- shm_id: What we got from shmget()

Detaches from this processes address space.

Example:

```
if (shmdt(p_shared) == -1)
{
        printf("Failed to detach from shared memory\n");
        return RTN_ERROR;
}
```



shmctl() is a multiuse function for performing OS calls to work with shared memory

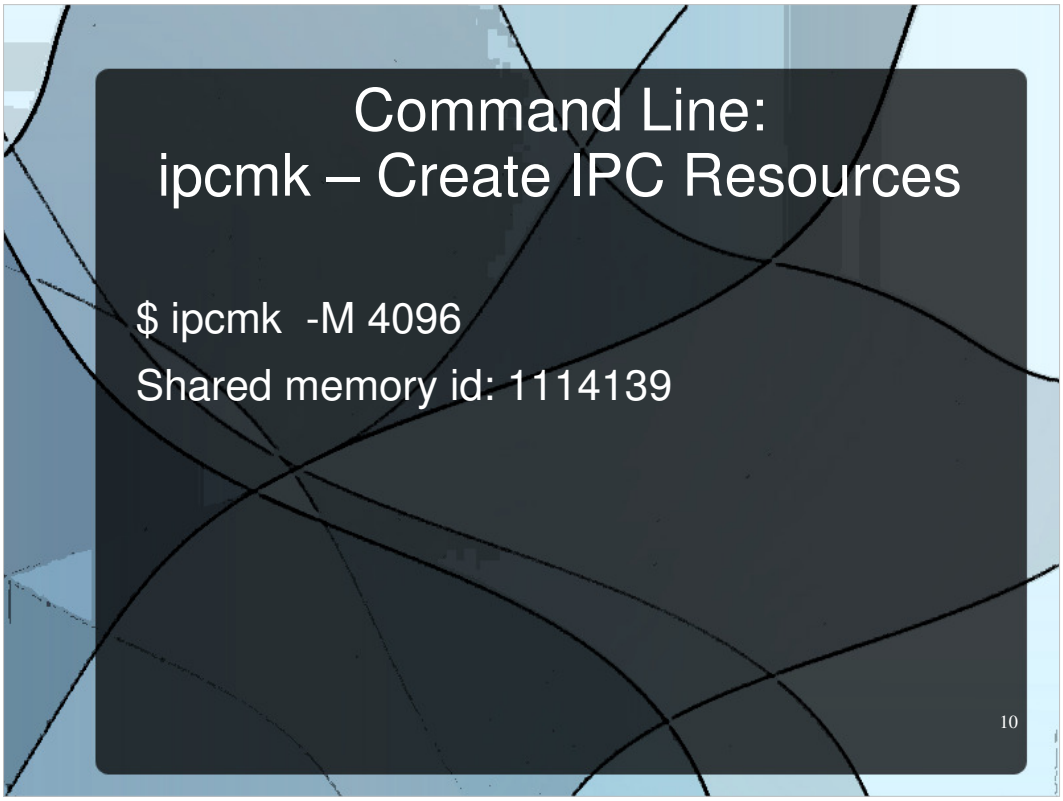
We must use shmctl() to tell the OS to completely remove the shared memory segment (not only from this process, but from physical memory too).

- shm_id: What we got from shmget()
- command: IPC_RMID to remove shared memory
- buf: Output buffer for other commands.

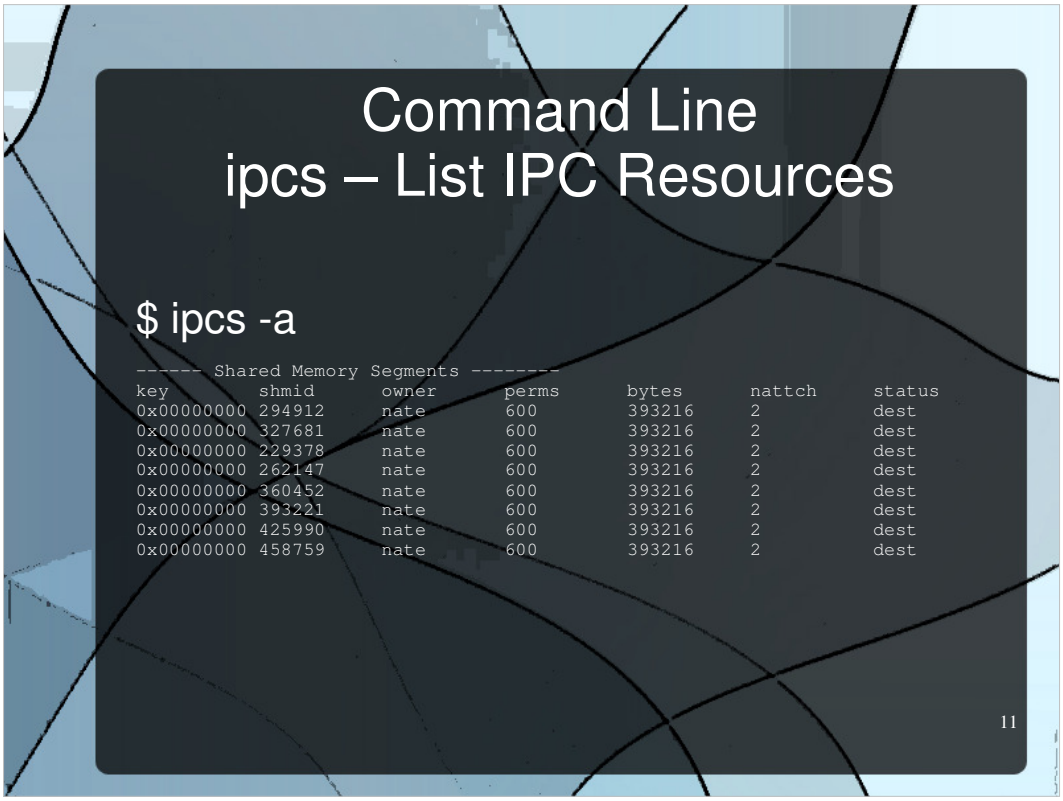
Tip: Using a flag, you can say ”Delete shared memory iff no processes are attached to it”

Example:

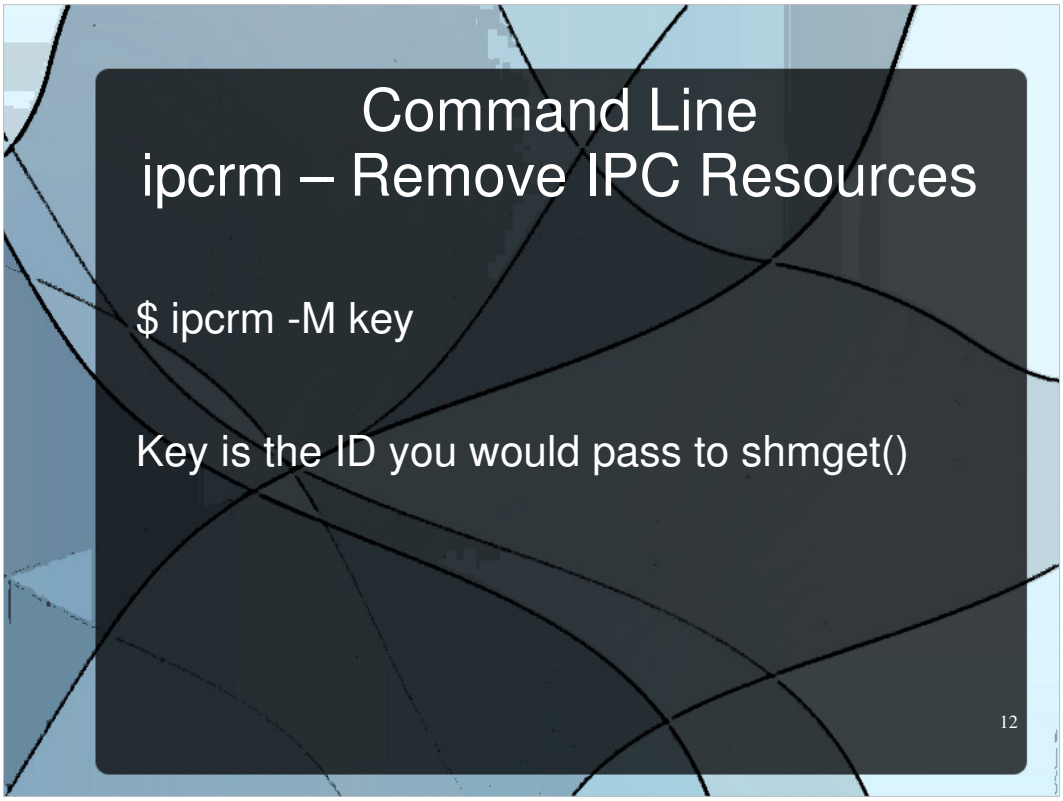
```
shmctl(shm_id, IPC_RMID, 0)
```




```
$ man ipcmk

IPCMK(1)                Linux Programmer's Manual                IPCMK(1)

NAME
       ipcmk - create various ipc resources

SYNOPSIS
       ipcmk [ -M size ] [ -p mode ]
       ipcmk [ -S nsems ] [ -p mode ]
       ipcmk [ -Q ] [ -p mode ]

DESCRIPTION
       ipcmk  allows you to create shared memory segments, message queues
       or semaphore arrays.

OPTIONS
       Resources may be specified as follows:

       -M size
              shared memory segment of size size

       -S nsems
              semaphore array with nsems elements

       -Q     message queue

       Other options

       -p mode
              permission for the resource (default is 0644)
```

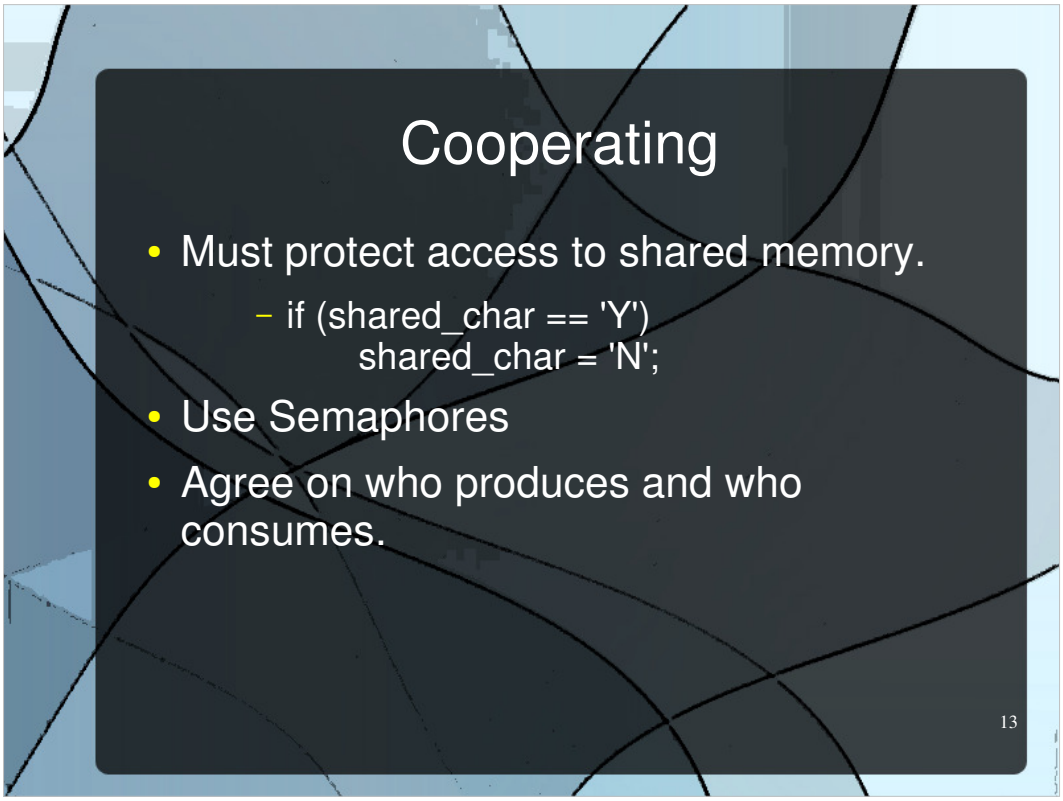



```
$ man ipcs
IPCS(1)                  Linux Programmer's Manual                 IPCS(1)

NAME
       ipcs - provide information on ipc facilities

SYNOPSIS
       ipcs [ -asmq ] [ -tclup ]
       ipcs [ -smq ] -i id
       ipcs -h

DESCRIPTION
       ipcs  provides  information  on  the  ipc facilities for which the
       calling process has read access.

       The -i option allows a specific resource id to be specified.  Only
       information on this id will be printed.

       Resources may be specified as follows:

       -m     shared memory segments

       -q     message queues

       -s     semaphore arrays

       -a     all (this is the default)

       The output format may be specified as follows:

       -t     time

       -p     pid

...
```



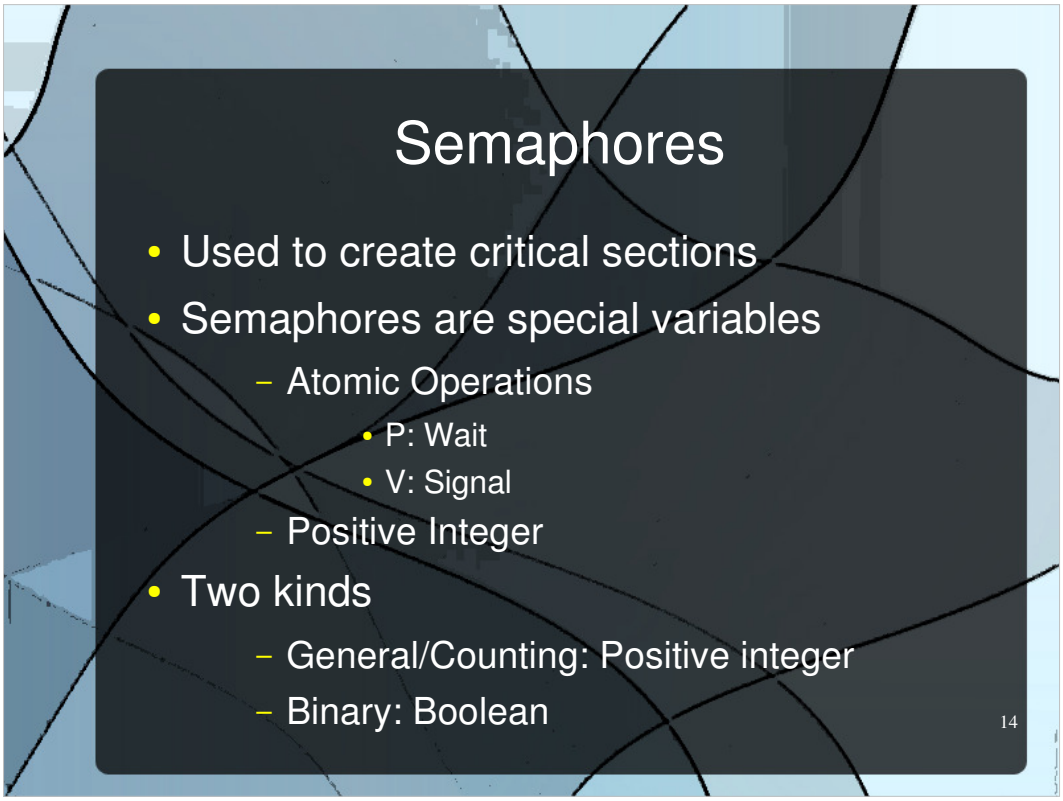
Useful when you're developing and your program failed to clean up.

```
$ man ipcrm
IPCRM(1)                Linux Programmer's Manual                IPCRM(1)

NAME
       ipcrm - remove a message queue, semaphore set or shared memory id

SYNOPSIS
       ipcrm [ -M key | -m id | -Q key | -q id | -S key | -s id ] ...

       deprecated usage

       ipcrm [ shm | msg | sem ] id ...

DESCRIPTION
       ipcrm  removes  System  V interprocess communication (IPC) objects
       and associated data structures  from  the  system.   In  order  to
       delete  such  objects,  you  must  be superuser, or the creator or
       owner of the object.

       System V IPC objects are of three types:  shared  memory,  message
       queues,  and semaphores.  Deletion of a message queue or semaphore
       object is immediate (regardless of whether any process still holds
       an IPC identifier for the object).  A shared memory object is only
       removed after  all  currently  attached  processes  have  detached
       (shmdt(2)) the object from their virtual address space.

       Two  syntax styles are supported.  The old Linux historical syntax
       specifies a three letter keyword indicating which class of  object
       is  to  be  deleted,  followed  by one or more IPC identifiers for
       objects of this type.

...
```



We should know why this code could fail.

I typically use semaphores to protect the shared memory critical section. (Semaphores also are provided by SysV IPC).

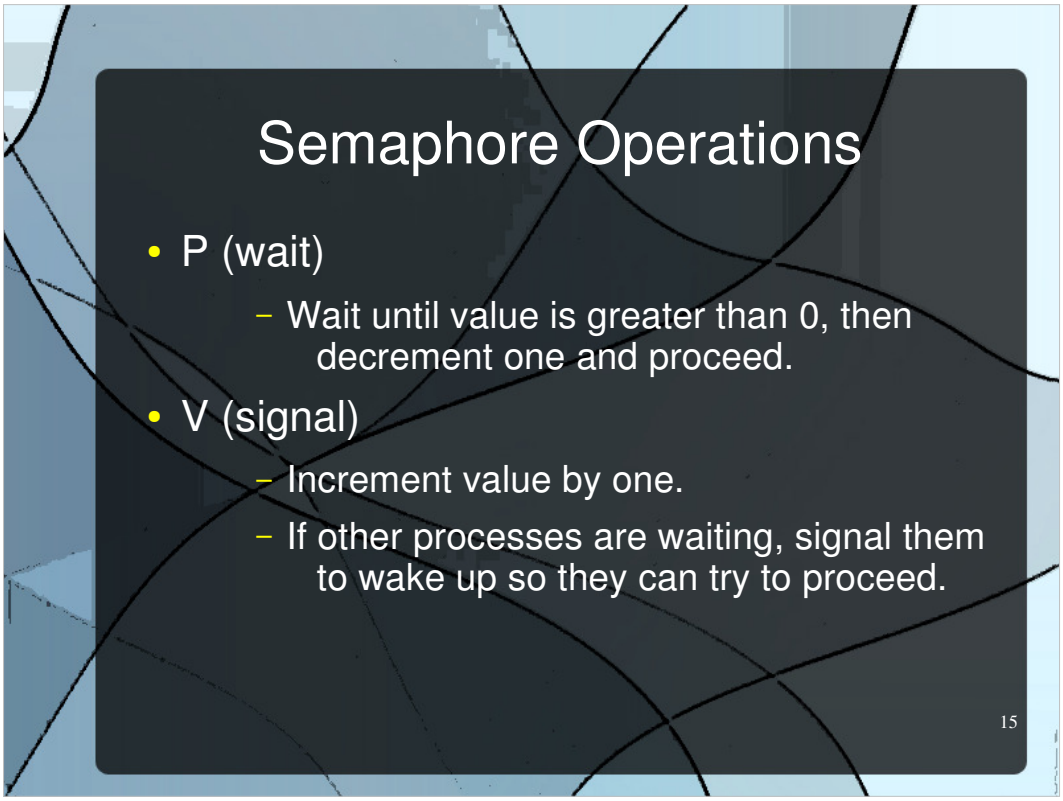


Use to control usage of a resource.

Critical section:
- Block of code that must be run by only one process at a time.

Made possible by an atomic instruction which reads and updates in one cycle/instruction. Or, sometimes it is simulated (i.e.: Milk Buying problem. See Dekler's Algorithm).

Note: Mutex is a special kind of binary semaphore. Mutexes are owned by process, usually used in threaded applications. They are limited to allowing only one process in the critical section at a time.

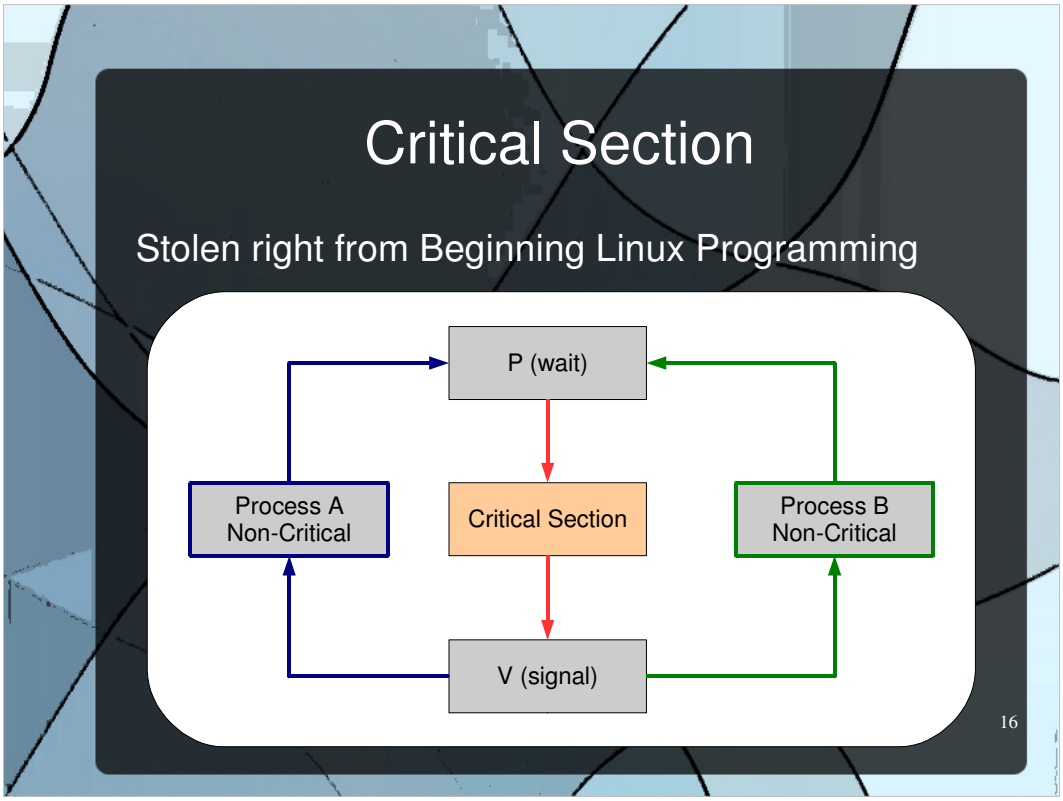


Sleeps during wait (better than busy wait).

Semaphore may lock again before a woken up process can decrement to 0.

Consider case where three processes all try to P a semaphore at once.

Need to prime semaphore to 1 (or number of processes allowed in critical section at once).



Blue: Process A's Loop
Green: Process B's Loop
Red: Critical Section

Notes:

- Time in critical section should be short
- Process sleeps while waiting
- There is an option to check and not block (I think)

Want to know more, check out:

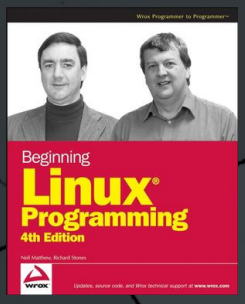
- sys/sem.h
- semctl()
- semget()
- semop()

Alternatives to Shared Memory

- mmap maps files (or devices) into memory
- Message queues
- Posix signals
- Network sockets (expensive)

Further Reading (My Sources)

- man svipc
  man shm_overview
  (For Ubuntu, install manpages-dev)
- Beginning Linux Programming, by Richard Stones and Neil Matthew
- Modern Operating Systems, by Andrew S. Tanenbaum
  (Previous OpSys Textbook)



Questions?