Category	System Call	Description
General	open ()	This system call either opens an already existing file or creates and opens a new file.
General	creat ()	Creates and opens a new file.
General	read ()	Reads the contents of the file into the required buffer.
General	write ()	Writes the contents of buffer into the file.
General	close ()	Closes the file descriptor.
General	stat ()	Provides information on the file. descriptors for reading and writing.
Named Pipes or Fifo	mknod ()	Creates a memory device file or special file to create FIFOs
Named Pipes or Fifo	mkfifo ()	Creates a new FIFO
Shared Memory	shmget ()	Creates a new shared memory segment or gets the identifier of the existing segment.
Shared Memory	shmat ()	Attaches the shared memory segment and makes the segment a part of the virtual memory of the calling process.
Shared Memory	shmdt ()	Detaches the shared memory segment.
Shared Memory	shmctl ()	Performs control operations for the shared memory. Few of the generic control operations for the shared memory are removing the shared memory segment (IPC_RMID), receiving the information of the shared memory (IPC_STAT) and updating new values of the existing shared memory (IPC_SET).
Message Queues	msgget ()	Creates a new message queue or accesses an already existing message queue and gets the handle or identifier to perform operations with regard to message queue, such as sending message/s to queue and receiving message/s from the queue.
Message Queues	msgsnd ()	Sends a message to the required message queue with the required identification number.
Message Queues	msgrcv ()	Receives a message from the message queue. By default, this is infinite wait operation, means the call will be blocked until it receives a message.
Message Queues	msgctl ()	Performs control operations for the message queue. Few of the generic control operations for the message queue are removing the message queue (IPC_RMID), receiving the information of the message queue (IPC_STAT) and updating new values of the existing message queue (IPC_SET).
Semaphores	semget ()	Creates a new semaphore or gets the identifier of the existing semaphore. Semaphores are used to perform synchronization between various IPCs working on the same object.
Semaphores	semop ()	Performs semaphore operations on semaphore values. The basic semaphore operations are either acquiring or releasing the lock on the semaphore.
Semaphores	semctl ()	Performs control operations for the semaphore. Few of the generic control operations for the semaphore are removing the semaphore (IPC_RMID), receiving the information of the semaphore (IPC_STAT) and updating new values of the existing semaphore (IPC_SET).
Signals	signal ()	Setting the disposition of the signal (signal number) and the signal handler. In other terms, registering the routine, which gets executed when that signal is raised.
Signals	sigaction ()	Same as signal(), setting the disposition of the signal i.e., performing certain action as per the registered signal handler after the receipt of the registered signal. This system call supports finer control over signal() such as blocking certain signals, restoring signal action to the default state after calling the signal handler, providing information such as consumed time of the user and the system, process id of sending process, etc.
Memory Mapping	mmap ()	Mapping files into the memory. Once mapped into the memory, accessing files is as easy as accessing data using addresses and also in this way, the call is not expensive as system calls.
Memory Mapping	munmap ()	Un-mapping the mapped files from the memory.