START

1. Header file Inclusion
2. prototyping of function
3. Initializing MUTEX
4. main function start
5. Initialise  'FD'  and create an unnamed pipe in variable 'a'
6. If pipe return - 1 then print error message
7. Else print the statement of successful creation
8. using signal pre-referencing of SIGCHLD system call with the help of function
9. using fork system call, create two process child and parent
10. check success and failure case of fork
11. use wait in parent and invokes the child process
12. child process execution

* check the ID’s  for parent/child/user/group
* Create 3 files with the help of create system call
* write the content of buffer using write system call for the file created
* Data is read from buffer using read system call for the file created
* set/initialize/acquire Semaphore for child process
* Create shared memory for 3 files(150 bytes)
* Transmit each file data into 3 different byte address (50 bytes each) by invoking three different processes.
* Child process terminated

1. Parent process executed

* Step1-perform the customized action
* Step2-Accept data from Pipe
* Step3-acquiring a lock on the specified MUTEX variable
* Step4-initializations of conditional variables
* Step5-creating the function for thread 1

1. locking the MUTEX using pthread\_mutex\_lock() system call
2. assigning the key value
3. getting the shared memory id using shmget() system call
4. getting shared memory pointer with the help of shmat() system call
5. on success returns the pointer to the attached segment and -1 on failure
6. if it success then reading the data from the shared memory
7. unlocking the MUTEX using using pthread\_mutex\_unlock() system call
8. exiting the thread using pthread\_exit(NULL) system call

* step 6- Repeating step 3 for thread 2
* step 7- Repeating step 3 for thread 3
* step 8- creating thread 1, 2 and 3 using pthread\_create() system call
* step 9- joining thread 1,2 and 3 using pthread\_join() system call

STOP