```
/*
                SHARING MEMORY BETWEEN PROCESSES

    In this example, we show how two processes can share a common
    portion of the memory. Recall that when a process forks, the
    new child process has an identical copy of the variables of
    the parent process. After fork the parent and child can update
    their own copies of the variables in their own way, since they
    dont actually share the variable. Here we show how they can
    share memory, so that when one updates it, the other can see
    the change.

*/

#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>   /*  This file is necessary for using shared
                           memory constructs
                       */

main()
{
        int shmid. status;
        int *a, *b;
        int i;

        /*
            The operating system keeps track of the set of shared memory
            segments. In order to acquire shared memory, we must first
            request the shared memory from the OS using the shmget()
            system call. The second parameter specifies the number of
            bytes of memory requested. shmget() returns a shared memory
            identifier (SHMID) which is an integer. Refer to the online
            man pages for details on the other two parameters of shmget()
        */
        shmid = shmget(IPC_PRIVATE, 2*sizeof(int), 0777|IPC_CREAT);
                        /* We request an array of two integers */

        /*
            After forking, the parent and child must "attach" the shared
            memory to its local data segment. This is done by the shmat()
            system call. shmat() takes the SHMID of the shared memory
            segment as input parameter and returns the address at which
            the segment has been attached. Thus shmat() returns a char
            pointer.
        */
        if (fork() == 0) {

                /* Child Process */

                /*  shmat() returns a char pointer which is typecast here
                    to int and the address is stored in the int pointer b. */
                b = (int *) shmat(shmid, 0, 0);
```

```
                for( i=0; i< 10; i++) {
                        sleep(1);
                        printf("\t\t\t Child reads: %d,%d\n",b[0],b[1]);
                }
                /* each process should "detach" itself from the
                   shared memory after it is used */

                shmdt(b);

        }
        else {

                /* Parent Process */

                /*  shmat() returns a char pointer which is typecast here
                    to int and the address is stored in the int pointer a.
                    Thus the memory locations a[0] and a[1] of the parent
                    are the same as the memory locations b[0] and b[1] of
                    the parent, since the memory is shared.
                */
                a = (int *) shmat(shmid, 0, 0);

                a[0] = 0; a[1] = 1;
                for( i=0; i< 10; i++) {
                        sleep(1);
                        a[0] = a[0] + a[1];
                        a[1] = a[0] + a[1];
                        printf("Parent writes: %d,%d\n",a[0],a[1]);
                }
                wait(&status);

                /* each process should "detach" itself from the
                   shared memory after it is used */

                shmdt(a);

                /* Child has exited, so parent process should delete
                   the cretaed shared memory. Unlike attach and detach,
                   which is to be done for each process separately,
                   deleting the shared memory has to be done by only
                   one process after making sure that noone else
                   will be using it
                 */

                shmctl(shmid, IPC_RMID, 0);
        }
}
```