1. **A simple C program to demonstrate use of pthread basic functions**  
   Please note that the below program may compile only with C compilers with pthread library.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  //Header file for sleep(). man 3 sleep for details.  #include <pthread.h>    // A normal C function that is executed as a thread  // when its name is specified in pthread\_create()  void\*myThreadFun(void\*vargp)  {      sleep(1);      printf("Printing from Thread \n");      returnNULL;  }    intmain()  {      pthread\_tthread\_id;      printf("Before Thread\n");      pthread\_create(&thread\_id, NULL, myThreadFun, NULL);      pthread\_join(thread\_id, NULL);      printf("After Thread\n");      exit(0);  } |

1. **A C program to show multiple threads with global and static variables**  
   As mentioned above, all threads share data segment. Global and static variables are stored in data segment. Therefore, they are shared by all threads. The following example program demonstrates the same.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <pthread.h>    // Let us create a global variable to change it in threads  intg = 0;    // The function to be executed by all threads  void\*myThreadFun(void\*vargp)  {      // Store the value argument passed to this thread      int\*myid = (int\*)vargp;        // Let us create a static variable to observe its changes      staticints = 0;        // Change static and global variables      ++s; ++g;        // Print the argument, static and global variables      printf("Thread ID: %d, Static: %d, Global: %d\n", \*myid, ++s, ++g);  }    intmain()  {      inti;      pthread\_ttid;        // Let us create three threads      for(i = 0; i< 3; i++)          pthread\_create(&tid, NULL, myThreadFun, (void\*)&tid);        pthread\_exit(NULL);      return0;  } |

3.

#include <stdio.h>

#include <pthread.h>

/\*thread function definition\*/

**void**\*threadFunction(**void**\*args)

{

**while**(1)

{

Sleep(2);

printf("I am threadFunction.\n");

}

}

**int**main()

{

/\*creating thread id\*/

pthread\_t id;

**int** ret;

/\*creating thread\*/

ret=pthread\_create(&id,NULL,&threadFunction,NULL);

**if**(ret==0){

printf("Thread created successfully.\n");

}

**else**{

printf("Thread not created.\n");

**return**0;/\*return from main\*/

}

**while**(1)

{

Sleep(2);

printf("I am main function.\n");

}

**return**0;

}

1. Write a C program to calculate the factorial value of a number by creating new thread. ( number should be passed through command line argument)
2. Implement the C program in which main program accepts list of integers. Main program prints the sum of odd numbers from the list of integers. A thread is created and it has to calculate the sum of even numbers from the same list and should be printed by the main thread.
3. Write a C program to display the student grade sheet. Thread1 should get the details of a student; Thread 2 should get the details of 5 subject marks. Main thread should calculate the grade point and prepare a grade sheet.