**Lab 8**

**Multithreading**

1. int pthread\_create(pthread\_t \*tid, pthread\_attr\_t \*att, void \*func, void \*args);

above function returns 0 in success case and non-zero in case of failure and this non zero is basically an error code.

The first paremeter is the thread id, and the second is NULL, and the third is function name which we want to execute, and the 4th argument is the arguments to the function.

1. int pthread\_join(pthread\_t tid, int \*\*status);
2. int pthread\_self()

returns the id of thread.

what is thread? It is flow of execution.

Example.c

int main()  
{

int tid;

int x=10;

int y=20;

th\_func(),

int z=x+y;

printf(“sum is %d\n”,z),

return 0;

}

void \*th-func()

{

int x=30;

int y=40;

int diff = x-y;

printf(“diff=%d”,diff);

}

What we need to do is to create a new thread and execute a specific task with it.

Example of creating thread.c

int main()  
{

int x=10;

int y=20;

pthread\_create(&tid, NULL, th\_func, NULL);

int z=x+y;

printf(“sum is %d\n”,z),

pthread\_join(tid, NULL);

return 0;

}

void \*th-func(void \*arg)

{

int x=30;

int y=40;

int diff = x-y;

printf(“diff=%d”,diff);

}

the main is basically thread 1 which is master thread, and all others created thread will be its children threads, if master thread terminates then all its children must be terminate this is known as cascading terminate. For continuity of thread execution, you need to wait for it using pthead\_join func.

Stack section is divided for threads, and local variables is duplicated.

Global section does not divide and global variable is accessible to all threads.