**DAY -14(14-11-2024)**

* commands
* Carriage return \r – cursor returns to the start of the same line we are working
* New line \n - cursor points to the start of the next line
* \b – backspace the values
* int x=10, y=20;

printf(“%d”, x, y); ---- o/p: 10

* int a=b=c=10;

a=b=c=50;

leads to the compile error we can’t assign values like this is C

**MULTI-THREADING**

* Command prompt is doing multi-processing which is parent and there are many processes(child) they are connected through the thread
* Thread is light weight process
* Thread is attaching the shell to process
* **CONCURRENCY**: perform multiple functions at a time or handle multiple operations within overlapping time frames. Necessities and benefits are below:
* Responsiveness –it allows applications to remain responsive to user actions even when performing long running tasks in the background
* Utilizing multi core processors
* Efficient resource utilization
* Scalability
* Improved throughput – it is amount of work an application can process in a given amount of time is often increased
* Concurrency in programming:

1. Multi – threading

* multiple threads within a single process are created to perform different parts of task or handle multiple task concurrently.
* Threads share the same memory space, which allows for efficient communication but requires careful synchronization to prevent race conditions.
* Race conditions if both operations perform on same data
* Each thread has its own stack,register and program counter.

1. Multi-processing

* Separate processes are created to handle different tasks concurrently
* Each process has its own memory space, which is safer but requires inter process communication methods like pipe or shared memory for collaboration.
* Eg : coco game

1. Asynchronous programming

* Non-blocking functions are used to handle task such as I/O operations without blocking the main thread, allowing other operations are perform concurrently
* Parallelism—executing multiple tasks truly simultaneously ,typically achieved by having multiple cpu core
* Concurrency – dealing with multiple task at once or having multiple tasks in progress
* Methods to achieve concurrency in c:

1. Multithreading using posix threads(pthreads)---create and manage threads within a process
2. Using fork() for multiprocessing – create separate process with individual memory spaces
3. Asynchronous I/O
4. Atomic operations--- thread safe operations
5. Semaphores – controls access to shared resources by multiple threads
6. Message queues

* Traditional view of a process :

Process =process context+code,data,stack

* Alternate view of process:

Process=thread+code,data and kernel context

* A process with multiple threads

Multiple threads can be associated with a process

1. Each thread has its own logical control flow(sequence of pc values)
2. Each thread shares the same code,data and kernal context
3. Each thread has its own thread id (TID)

* THREADS vs PROCESSES:

How threads and processes are similarly:

1. Each has its own logical control flow
2. Can run concurrently
3. Context switched

How these different:

1. Thread share code ,data ,process donot

Thread shares memory space with other threads in process vs process has its own memory space

* POSIX threads(pthreads):

1. Creating and reaping threads

* Pthread\_create,pthread\_join

1. Determing your thread id

* Pthread\_self

1. Teriminating threads

* pthread\_cancel,pthread\_exit
* exit[terminates all threads],return[terminate current threads]

1. Synchronizing access to shared variables

* Pthread\_mutex\_init,pthread\_mutex\_[un]lock
* Pthread\_cond\_init,pthread\_cond\_[timid]wait
* On success, pthread\_create() returns 0; on error, it returns an error number, and the contents of \*thread are undefined.
* Use #include<pthread.h> and while compling use gcc filename -lpthread
* Pthread\_join means main have to wait for the function to complete execution
* If you use sleep() the main donot have to wait for function to complete the execution ,main have to wait for only the number of seconds sleep given
* programs for functions using pointer:
* Main()

{

Int a=10;

Int arr[2]={10,20};

Fun1(&a); //here we pass address a so that it can be used in fun1

Fun2(arr);//base address of the arr

Fun3(&arr);

}

Fun1(int \*a) // we catch the value

{

a=a+3;

}

Fun2(int a[] or int \*a)

{

a[1]=2;

}

Fun3(int \*\*b)

{

}

Program for threads:

/\*

thread demo

\*/

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#define ITNERS 1000000

void printHello();

int main()

{

pthread\_t tid;

int status=0;

printf("\nIn the Main Function\n");

printf("\nI am Main Program/Process/MainThread\n");

status = pthread\_create(&tid,NULL,printHello,NULL);

// pthread\_join(tid,NULL);

sleep(1);

printf("\nAfter completing (Main)\n");

return 0;

}

void printHello()

{

int i;

printf("\nHello World\n");

for(i=0;i<ITNERS;i++)

printf("\n\ti=%d",i);

}

* If I want to pass structure convert into void star type and assign it and catch it and execute it
* In threads pthread\_create(&tid,NULL,printHello,NULL);

Thread attribute functionname arguments of the function

Here arguments of the function is can be NULL and the (void\*) arguments

* How to efficient access the linear search –ans: threads(multi threading)
* Mutex: its locking of the first thread
* Pthread\_mutex\_lock lock //name of the mutex lock
* Pthread\_mutex\_init(&lock,NULL) // initialize the mutex
* Pthread\_mutex\_lock(&lock) //lock the mutex
* Pthread\_mutex\_unlock(&lock) //unlock
* Mutex cannot be used in the different functions,here we can use the different threads for different functions.in mutex also we use different threads for same function.
* This mutex is used only if many threads are used same shared memory
* This mutex gives the correct value
* Pthread\_mutex\_destroy(&lock); 🡺 mutex lock must destroyed
* Why the mutex lock is destroyed?
* In do-while loop why the semi colon at the end of the while loop 🡺 because do it execute only one time and from the next iteration onwards the while is executed so the complier have to recognise it as a single statement and same is applicable for unions and structures.