NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL

DEPARTMENT OF INFORMATION TECHNOLOGY

**IT 301 Parallel Computing LAB 4**

24th August 2021

Faculty: Dr. Geetha V

---------------------------------------------------------------------------------------------------------------------

**NAME: BHUVANESWAR DHARMASIVAM**

**ROLL NO: 191IT107**

**1. Understanding concept of schedule. Write the observation using schedule (static, 5), schedule (dynamic, 5) and schedule (guided, 5) [Marks: 1+1+1=3]**

#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

int main (void) {

int i;

#pragma omp parallel num\_threads(4)

{

#pragma omp for schedule(guided,5) private(i)

for(i=0;i<27;i++)

{

printf("tid=%d, i=%d \n",omp\_get\_thread\_num(),i);

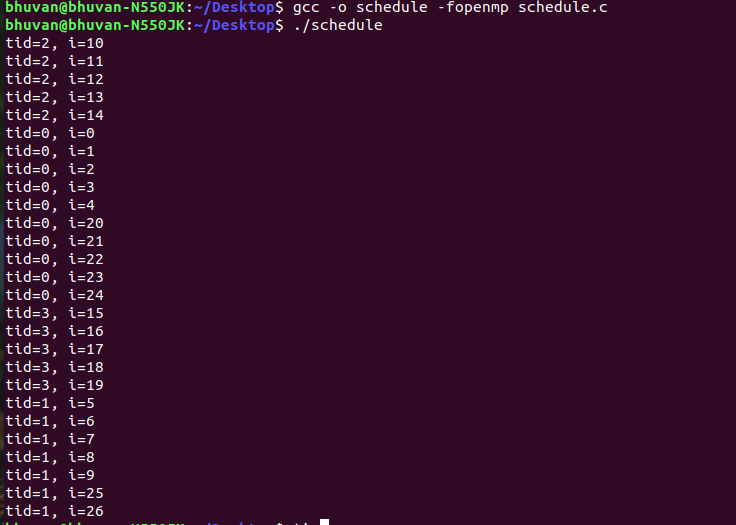
}

}

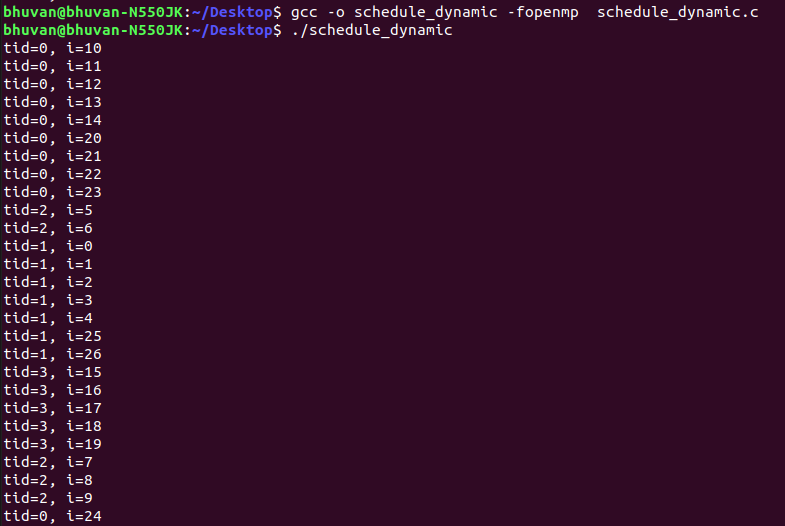
return 0;

}

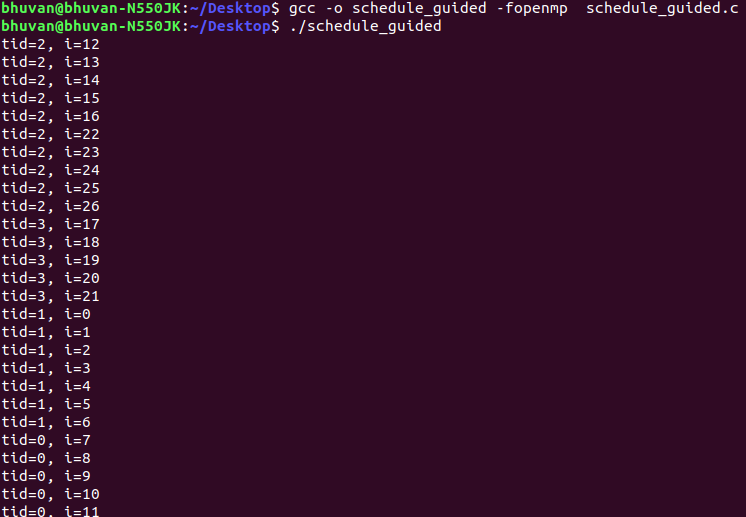
**static**



**dynamic**



**guided**



**Observation**:

In **static,** iterations are divides equally to different threads in chunk sizes and executed. In **Dynamic**, iterations are assigned to each thread in chunk sizes and last thread might have different. In **Guided**, thread executes the chunk of iteration and then requests another chunk, until all iterations are complete

**2. Execute following code and observe the working of threadprivate directive and copyin clause:**

#include<stdio.h>

#include<omp.h>

int tid,x;

#pragma omp threadprivate(x,tid)

void main()

{

x=10;

#pragma omp parallel num\_threads(4) copyin(x)

{

tid=omp\_get\_thread\_num();

#pragma omp master

{

printf("Parallel Region 1 \n");

x=x+1;

}

#pragma omp barrier

if(tid==1)

x=x+2;

printf("Thread % d Value of x is %d\n",tid,x);

}//#pragma omp barrier

#pragma omp parallel num\_threads(4)

{

#pragma omp master

{

printf("Parallel Region 2 \n");

}

#pragma omp barrier

printf("Thread %d Value of x is %d\n",tid,x);

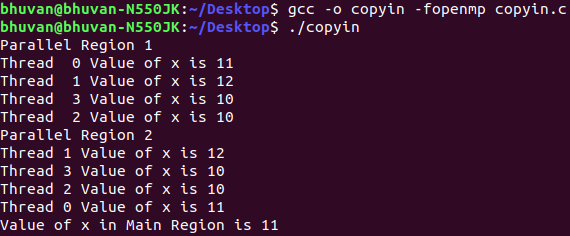
}

printf("Value of x in Main Region is %d\n",x);

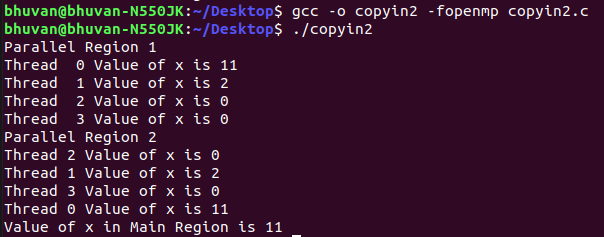
}

**Do the following: [Marks: 1+1=2]**

**a. Remove copyin clause and check the output.**

****

**b. Remove copyin clause and initialize x globally.**

**Note the observation about threadprivate directive and copyin** **clause.**

**Observation**: Threadprivate allows each thread is allowed to have its own temporary view of the shared memory. And copyin allows threads to access the master thread's value, for a [threadprivate](https://docs.microsoft.com/en-us/cpp/parallel/openmp/reference/openmp-directives?view=msvc-160" \l "threadprivate) variable.

**-----------------------------------------------------------------------------------**

**3. Learn the concept of firstprivate() and threadprivate()**

#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

int count=0;

#pragma omp threadprivate(count)

int main (void) {

int x=10, y=20,a[10],b[10],c[10],i;

//int count=0;

for(i=0;i<10;i++)

b[i]=c[i]=i;

printf("1. count=%d\n",count);

#pragma omp parallel num\_threads(2) copyin(count)

{

#pragma omp for schedule(static,5) firstprivate(x)

for(i=0;i<10;i++)

{

int tid1=omp\_get\_thread\_num();

a[i]=b[i]+c[i];

count++;

x++;

printf("tid=%d,a[%d]=%d, count=%d x=%d\n",tid1,i,a[i],count,x);

}

#pragma omp barrier

printf("2. before copyprivate count=%d x=%d tid=%d\n",count,x,omp\_get\_thread\_num());

#pragma omp single copyprivate(count)

{

count=count+20;

}

printf("3. after copyprivate count=%d x=%d tid=%d\n",count,x,omp\_get\_thread\_num());

#pragma omp for schedule(static,5) firstprivate(x)

for(i=0;i<10;i++)

{

int tid1=omp\_get\_thread\_num();

a[i]=b[i]\*c[i];

count++;

x++;

printf("tid=%d,a[%d]=%d, count=%d, x=%d\n",tid1,i,a[i],count,x);

}

}

#pragma omp barrier

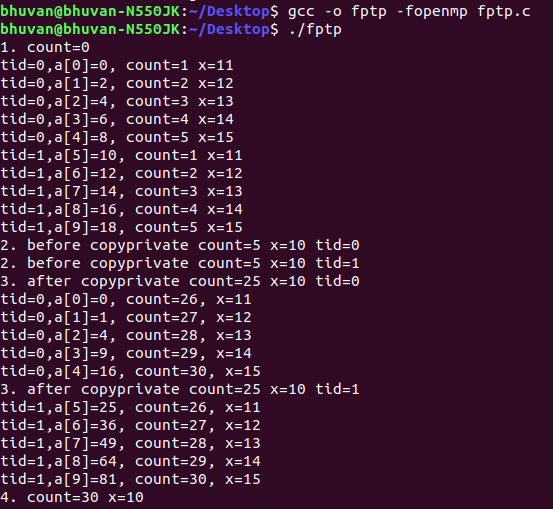
printf("4. count=%d x=%d\n",count,x);

printf("\n");

return 0;

}

**Analyse the results for variable count and x. write your observation [Marks: 1+1=2]**

****

**Observation:** Threadprivate allows each thread is allowed to have its own

temporary view of the shared memory. And copyprivaet broadcasts a value from the data environment of one implicit task to the data environments of the other implicit tasks belonging to the parallel region.

**---------------------------------------------------------------------------------------------**

**4. Program to understand the concept of collapse()**

#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

int main (void) {

int i,j;

#pragma omp parallel

{

#pragma omp for schedule(static,3) private(i,j) collapse(2)

for(i=0;i<6;i++)

for(j=0;j<5;j++)

{

int tid2=omp\_get\_thread\_num();

printf("tid=%d, i=%d j=%d\n",omp\_get\_thread\_num(),i,j);

}

}

return 0;

}

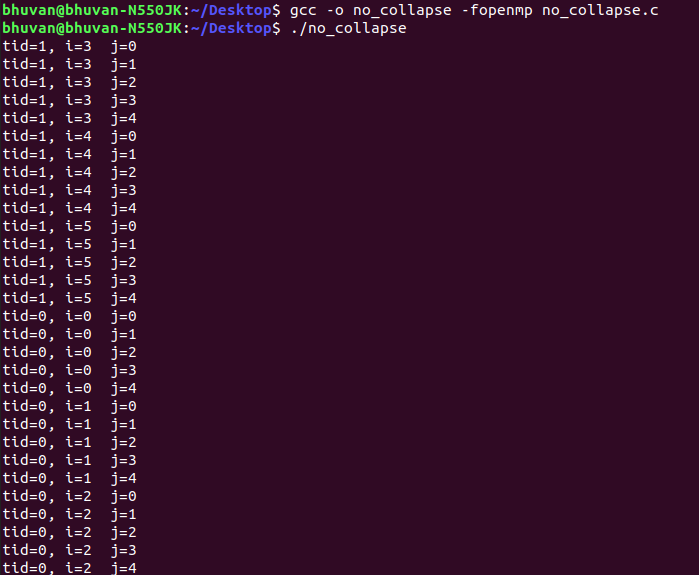
**Consider three for loops and check the result with no collapse(), collapse(2) and collapse(3). [1+1+1=3 Marks]**

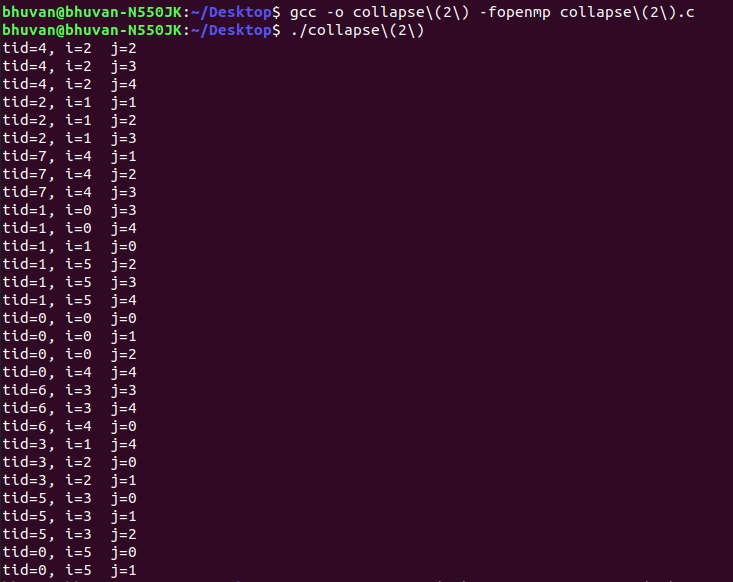
**Observation**: It increases the total number of iterations that

will be partitioned across the available number of OMP threads by reducing the granularity of work to be done by each

thread.

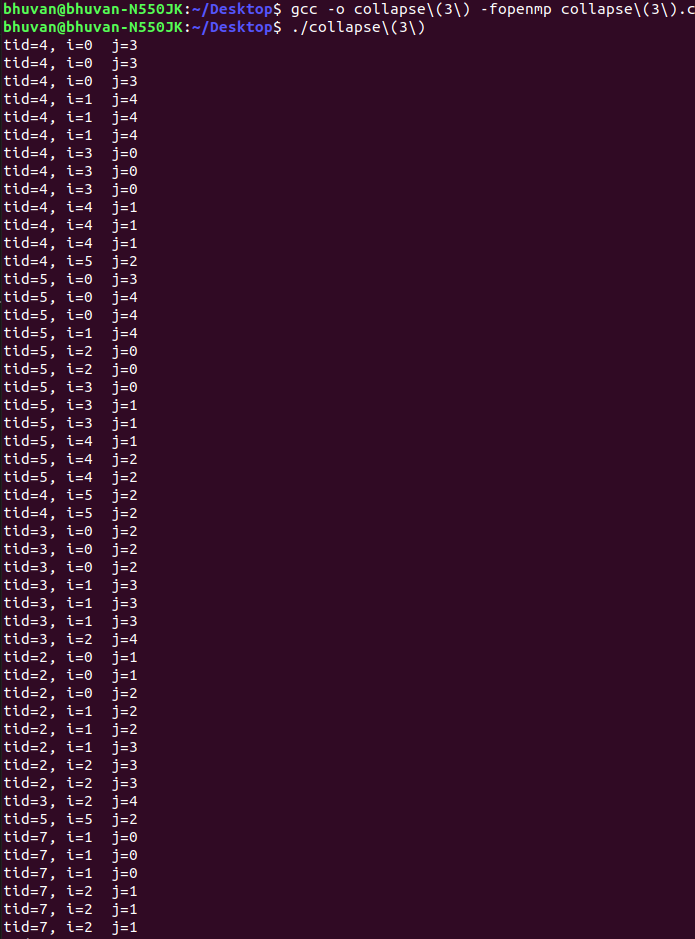
**no collapse()**

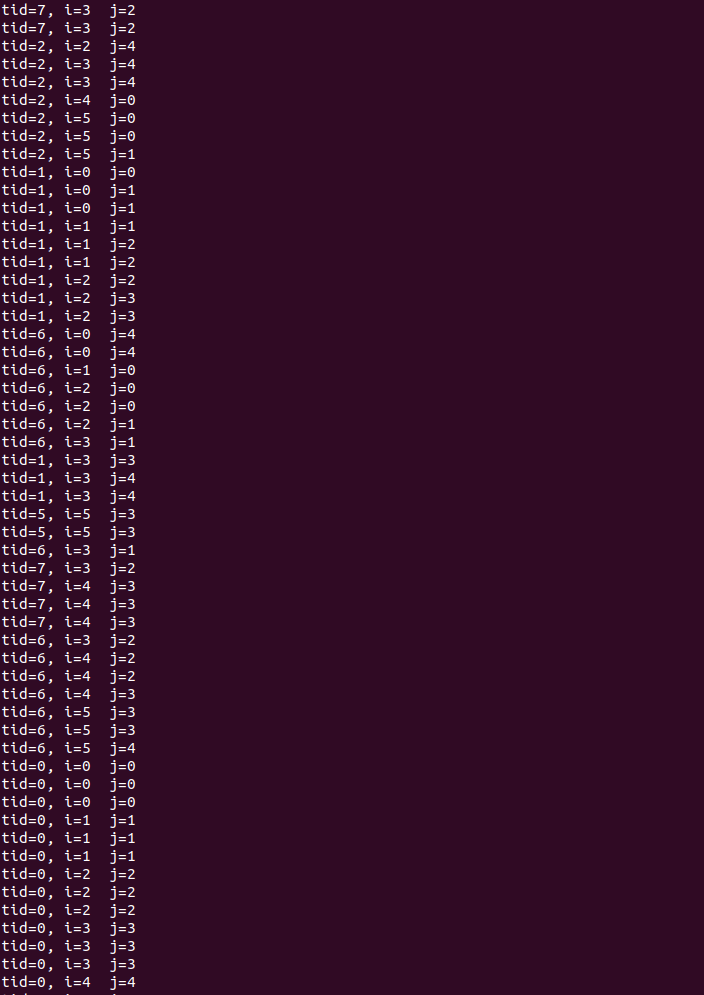
****

****

**collapse(2)**

**collapse(3)**

****

****

**-------------------------------------------------------------------------------------------**