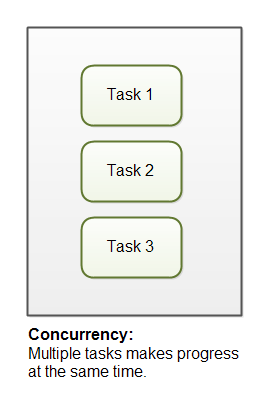
QUESTION 1:DIFFERENCE BETWEEN CONCURRENCY AND PARALLELISM?

ANSWER:

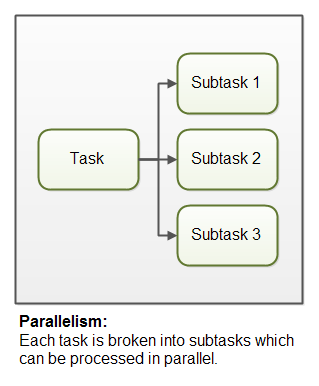
Concurrency

Concurrency means that an application is making progress on more than one task at the same time (concurrently). Well, if the computer only has one CPU the application may not make progress on more than one task at *exactly the same time*, but more than one task is being processed at a time inside the application. It does not completely finish one task before it begins the next.



Parallelism

Parallelism means that an application splits its tasks up into smaller subtasks which can be processed in parallel, for instance on multiple CPUs at the exact same time. Parallelism on the other hand, is related to how an application handles each individual task. An application may process the task serially from start to end, or split the task up into subtasks which can be completed in parallel.



QUESTION 2:HOW TO CHANGE TOP REFRESH TIME?

ANSWER:

* Start “top” command.
* Press “d”.
* Change delay time to 10 sec from 3 sec.
* Press shift + w. Notice message ‘Wrote configuration to ‘/home/wahaj/.toprs’.
* Quit “top”.
* Start “top” again.
* Delay is changed to 10 sec.

QUESTION 3:FIND SIGNAL NUMBER TO STOP A PROCESS USING KILL COMMAND?

ANSWER:

19 signal number is used to stop a process using kill command or SIG\_STOP is used.

* KILL\_19
* KILL\_SIGSTOP

QUESTION 4:USING FORK CALL SYSTEM PRINT HELLO INSIDE A LOOP.WRITE A CODE WHICH HAVE LOOP FROM I=0 TO I<=4.OUTSIDE A LOOP PRINT EXIT.ALSO DRAW TREE DIAGRAM?

ANSWER:The loop will display “hello” 30 times and “exit” 16 times.

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

int main()

{

for (int i = 0; i <= 4; i++)

{

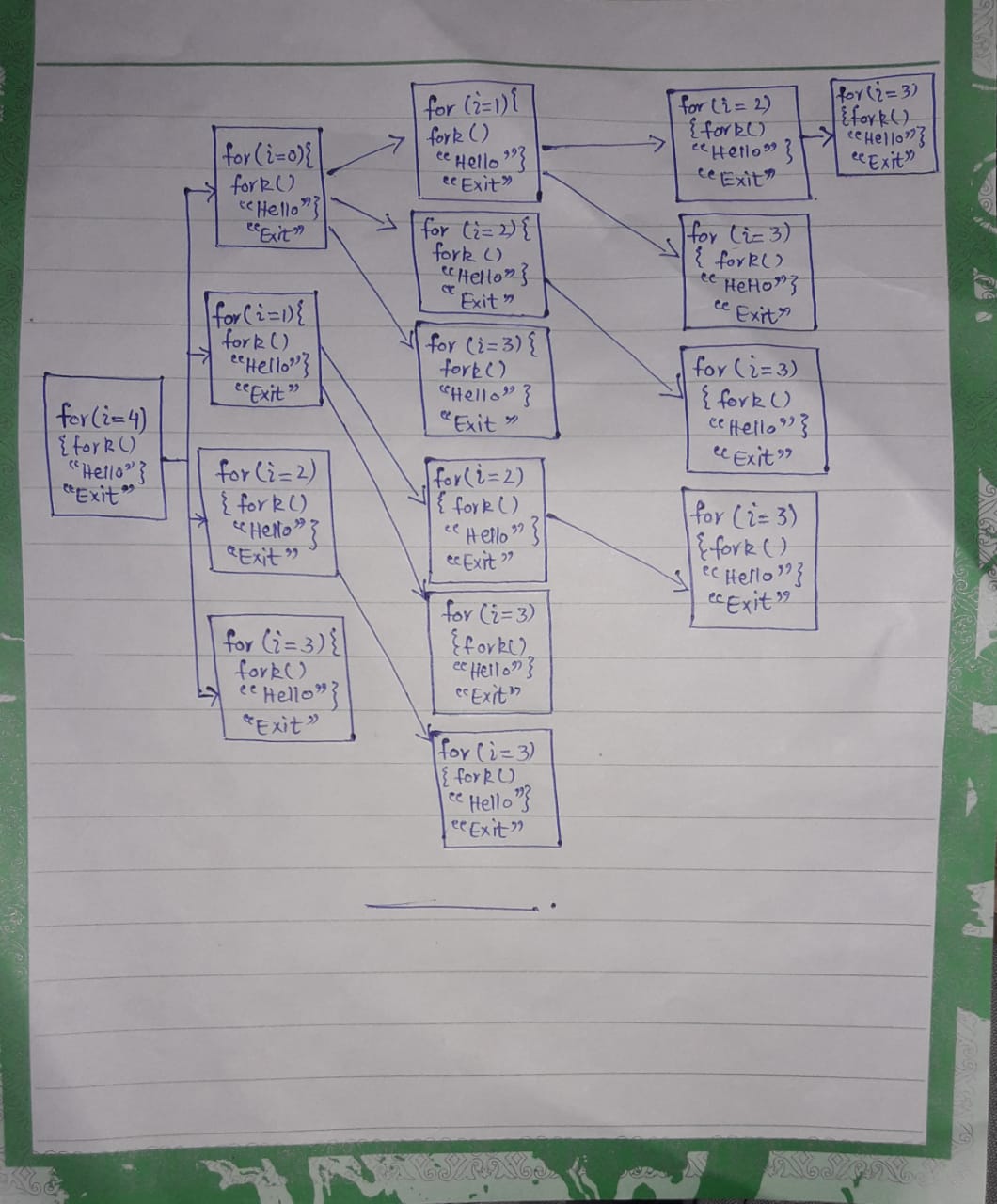
fork();

printf(“hello”);

}

printf(“Exit”);

}



QUESTION 5:CREATE ARRAY OF 1000 NO. AND INITIATE ARRAY WITH INDEX NO.CREATE 10 CHILD PROCESS. EACH CHILD PROCESS ADDS 100NO AND PARENT AT THE END ADDS UP THOSE 10 SUM VALUES AND GIVE OUTPUT?

ANSWER:

BY USING PIPE:

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/wait.h>

int main(){

int a[1000],a1[10];

int s1[2],s2[2],s3[2],s4[2],s5[2],s6[2],s7[2],s8[2],s9[2],s10[2];

int v1=0,v2=0,v3=0,v4=0,v5=0,v6=0,v7=0,v8=0,v9=0,v10=0,sum=0;

for(int i=0;i<1000;i++){

a[i]=i;

}

if(pipe(s1)==-1){printf("Pipe failed 1\n");

return 0;}

if(pipe(s2)==-1){printf("Pipe failed 2\n");

return 0;}

if(pipe(s3)==-1){printf("Pipe failed 3\n");

return 0;}

if(pipe(s4)==-1){printf("Pipe failed 4\n");

return 0;}

if(pipe(s5)==-1){printf("Pipe failed 5\n");

return 0;}

if(pipe(s6)==-1){printf("Pipe failed 6\n");

return 0;}

if(pipe(s7)==-1){printf("Pipe failed 7\n");

return 0;}

if(pipe(s8)==-1){printf("Pipe failed 8\n");

return 0;}

if(pipe(s9)==-1){printf("Pipe failed 9\n");

return 0;}

if(pipe(s10)==-1){printf("Pipe failed 10\n");

return 0;}

int cpid1=fork();

if(cpid1==0)

{

for(int i1=0;i1<100;i1++){

v1+=a[i1];

}

write(s1[1],&v1,sizeof(int));

close(s1[1]);

exit(0);

}

else{

wait(NULL);

int cpid2=fork();

if(cpid2==0){

for(int i2=100;i2<200;i2++){

v2+=a[i2];

}

write(s2[1],&v2,sizeof(int));

close(s2[1]);

exit(0);

}

else{

wait(NULL);

int cpid3=fork();

if(cpid3==0){

for(int i3=200;i3<300;i3++){

v3+=a[i3];

}

write(s3[1],&v3,sizeof(int));

close(s3[1]);

exit(0);

}

else{

wait(NULL);

int cpid4=fork();

if(cpid4==0){

for(int i4=300;i4<400;i4++){

v4+=a[i4];

}

write(s4[1],&v4,sizeof(int));

close(s4[1]);

exit(0);

}

else{

wait(NULL);

int cpid5=fork();

if(cpid5==0){

for(int i5=400;i5<500;i5++){

v5+=a[i5];

}

write(s5[1],&v5,sizeof(int));

close(s5[1]);

exit(0);

}

else{

wait(NULL);

int cpid6=fork();

if(cpid6==0){

for(int i6=500;i6<600;i6++){

v6+=a[i6];

}

write(s6[1],&v6,sizeof(int));

close(s6[1]);

exit(0);

}

else{

wait(NULL);

int cpid7=fork();

if(cpid7==0){

for(int i7=600;i7<700;i7++){

v7+=a[i7];

}

write(s7[1],&v7,sizeof(int));

close(s7[1]);

exit(0);

}

else{

wait(NULL);

int cpid8=fork();

if(cpid8==0){

for(int i8=700;i8<800;i8++){

v8+=a[i8];

}

write(s8[1],&v8,sizeof(int));

close(s8[1]);

exit(0);

}

else{

wait(NULL);

int cpid9=fork();

if(cpid9==0){

for(int i9=800;i9<900;i9++){

v9+=a[i9];

}

write(s9[1],&v9,sizeof(int));

close(s9[1]);

exit(0);

}

else{

wait(NULL);

for(int i10=900;i10<1000;i10++){

v10+=a[i10];

}

write(s10[1],&v10,sizeof(int));

close(s10[1]);

}

}

}

}

}

}

}

}

}

read(s1[0],&v1,sizeof(int));

read(s2[0],&v2,sizeof(int));

read(s3[0],&v3,sizeof(int));

read(s4[0],&v4,sizeof(int));

read(s5[0],&v5,sizeof(int));

read(s6[0],&v6,sizeof(int));

read(s7[0],&v7,sizeof(int));

read(s8[0],&v8,sizeof(int));

read(s9[0],&v9,sizeof(int));

read(s10[0],&v10,sizeof(int));

sum+=v1;

sum+=v2;

sum+=v3;

sum+=v4;

sum+=v5;

sum+=v6;

sum+=v7;

sum+=v8;

sum+=v9;

sum+=v10;

printf("Sum = %d",sum);

printf("\n\n");

close(s1[0]);

close(s2[0]);

close(s3[0]);

close(s4[0]);

close(s5[0]);

close(s6[0]);

close(s7[0]);

close(s8[0]);

close(s9[0]);

close(s10[0]);

return 0;

}