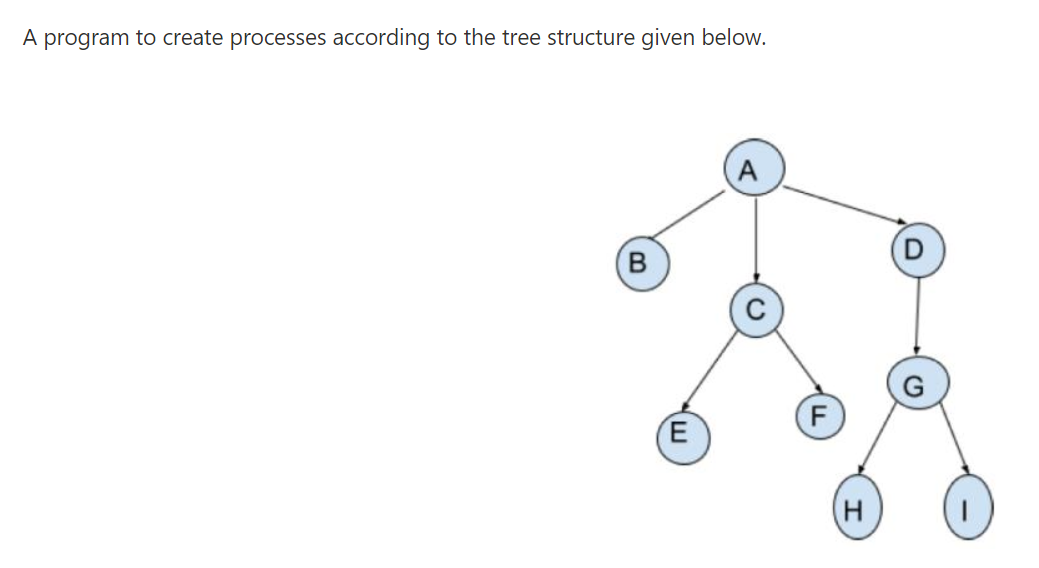
**S ABHISHEK AM.EN.U4CSE19147**

**OPERATING SYSTEM LAB 5**



#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

pid\_t pid = getpid();

pid\_t pptd = getppid();

printf("Label -> A PID -> %d PPID -> %d\n", getpid(),getppid());

if(fork())

{

wait(NULL);

if(fork())

{

wait(NULL);

if(!fork())

{

printf("Label -> D PID -> %d PPID -> %d\n", getpid(),getppid());

if(!fork())

{

printf("Label -> G PID -> %d PPID -> %d\n", getpid(),getppid());

if(fork())

{

wait(NULL);

if(!fork())

{

printf("Label -> I PID -> %d PPID -> %d\n", getpid(),getppid());

}

else

{

wait(NULL);

}

}

else

{

printf("Label -> H PID -> %d PPID -> %d\n", getpid(),getppid());

}

}

else

{

wait(NULL);

}

}

else

{

wait(NULL);

}

}

else

{

printf("Label -> C PID -> %d PPID -> %d\n", getpid(),getppid());

if(fork())

{

wait(NULL);

if(!fork())

{

printf("Label -> F PID -> %d PPID -> %d\n", getpid(),getppid());

}

else{

wait(NULL);

}

}

else{

printf("Label -> E PID -> %d PPID -> %d\n", getpid(),getppid());

}

}

}

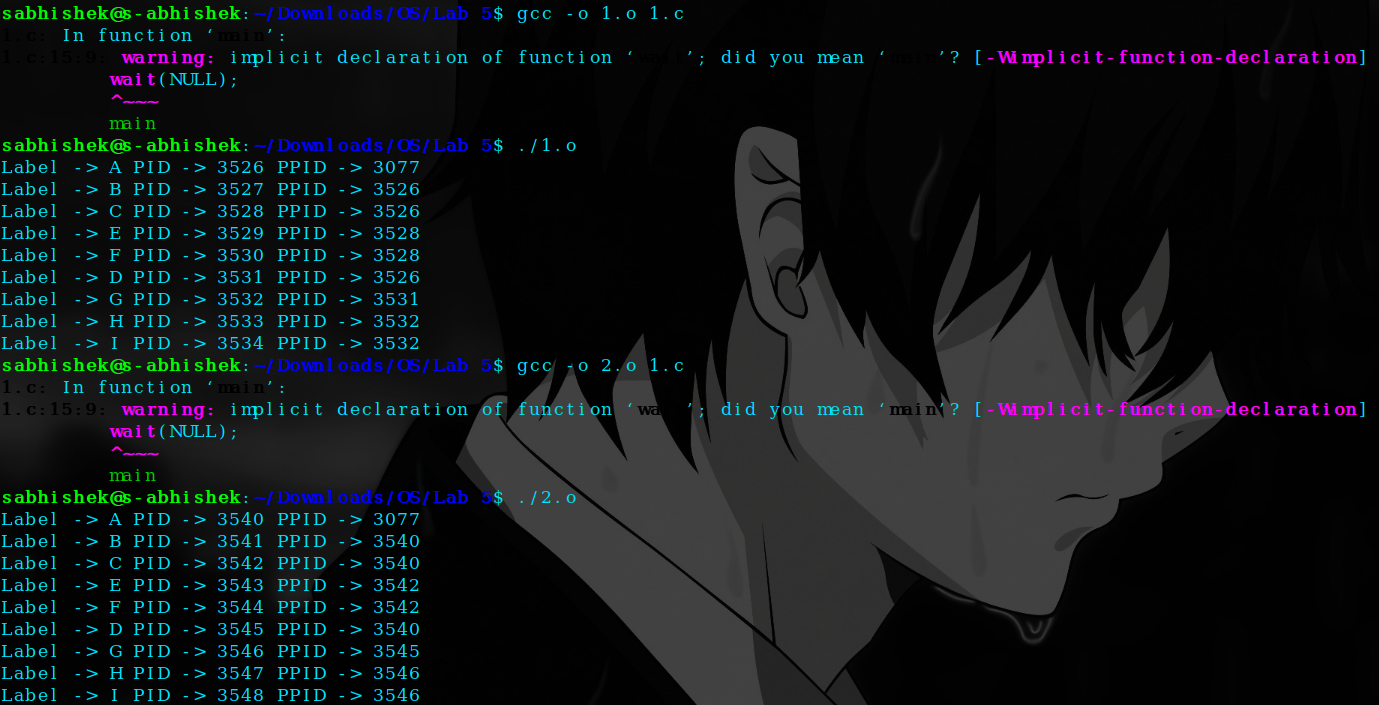
else {

printf("Label -> B PID -> %d PPID -> %d\n", getpid(),getppid());

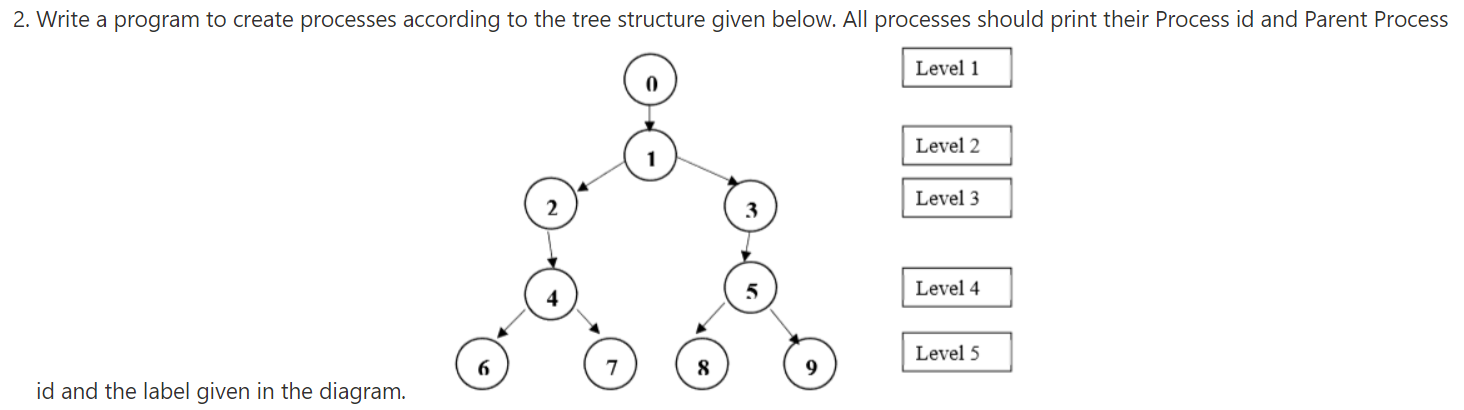
}

return 0;

}



* Even though the program is compiled several times, the order of the execution of the process is same.
* The order of execution is A-B-C-E-F-D-G-H-I.
* Only the process ID ( Parent ID and Child ID ) changes.
* Parent Process calls the fork() which creates the child process and using wait() system call the parent process waits until the child terminates.



#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

pid\_t pid = getpid();

pid\_t pptd = getppid();

printf("Label -> 0 PID -> %d PPID -> %d\n", getpid(),getppid());

if(!fork())

{

printf("Label -> 1 PID -> %d PPID -> %d\n", getpid(),getppid());

if(fork())

{

wait(NULL);

if(!fork())

{

printf("Label -> 3 PID -> %d PPID -> %d\n", getpid(),getppid());

if(!fork())

{

printf("Label -> 5 PID -> %d PPID -> %d\n", getpid(),getppid());

if(fork())

{

wait(NULL);

if(!fork())

{

printf("Label -> 9 PID -> %d PPID -> %d\n", getpid(),getppid());

}

else

{

wait(NULL);

}

}

else

{

printf("Label -> 8 PID -> %d PPID -> %d\n", getpid(),getppid());

}

}

else

{

wait(NULL);

}

}

else

{

wait(NULL);

}

}

else

{

printf("Label -> 2 PID -> %d PPID -> %d\n", getpid(),getppid());

if(!fork())

{

printf("Label -> 4 PID -> %d PPID -> %d\n", getpid(),getppid());

if(fork())

{

wait(NULL);

if(!fork())

{

printf("Label -> 7 PID -> %d PPID -> %d\n", getpid(),getppid());

}

else

{

wait(NULL);

}

}

else

{

printf("Label -> 6 PID -> %d PPID -> %d\n", getpid(),getppid());

}

}

else

{

wait(NULL);

}

}

}

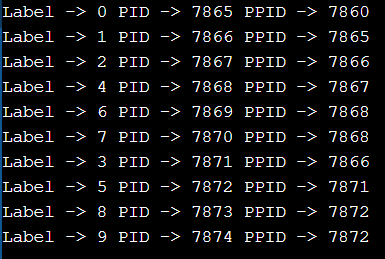
else

{

wait(NULL);

}

}



**3. Write a program to find the area and perimeter of circle and square.**

**Create separate processes to perform the calculation of circle and square.**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

if(fork())

{

wait(NULL);

int radius;

printf("\n\nEnter the Radius of the Circle : ");

scanf("%d",&radius);

printf("Perimeter of the Circle : %0.2f",2\*3.14\*radius);

printf("\nArea of the Circle : %0.2f",3.14\*radius\*radius);

}

else

{

float side;

printf("Enter length of side of square : ");

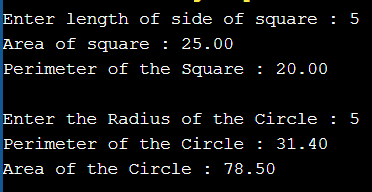
scanf("%f", &side);

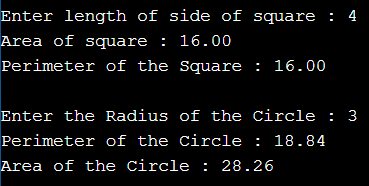
printf("Area of square : %0.2f",side\*side);

printf("\nPerimeter of the Square : %0.2f",4\*side);

}

}

****



**4. Modify the above program as follows: The parent process should create two children.**

**[User enters Value of variable ‘a’ only once].**

**The first child finds the area and perimeter of a circle with radius ‘a’.**

**The Second child finds the area and perimeter of square with side ‘a’.**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

float num;

printf("Enter the Number : ");

scanf("%f",&num);

pid\_t pid = getpid();

pid\_t pptd = getppid();

if(fork())

{

if(!fork())

{

printf("\nChild -> 2 PID -> %d PPID -> %d\n", getpid(),getppid());

printf("\nArea of square : %0.2f",num\*num);

printf("\nPerimeter of the Square : %0.2f",4\*num);

}

}

else

{

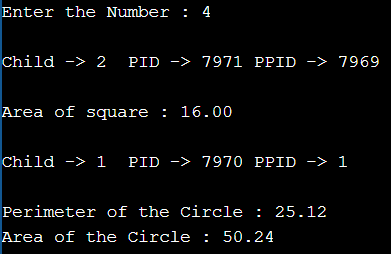
printf("\nChild -> 1 PID -> %d PPID -> %d\n", getpid(),getppid());

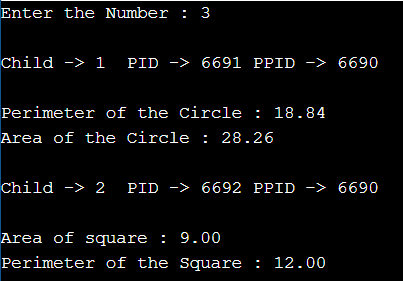
printf("\nPerimeter of the Circle : %0.2f",2\*3.14\*num);

printf("\nArea of the Circle : %0.2f\n",3.14\*num\*num);

}

}





**5. Modify the previous program to make the parent process wait until the completion of its children. [Hint. Use wait() system call]**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

float num;

printf("Enter the Number : ");

scanf("%f",&num);

pid\_t pid = getpid();

pid\_t pptd = getppid();

if(fork())

{

wait(NULL);

if(!fork())

{

printf("\nChild -> 2 PID -> %d PPID -> %d\n", getpid(),getppid());

printf("\nArea of square : %0.2f",num\*num);

printf("\nPerimeter of the Square : %0.2f",4\*num);

}

else

{

wait(NULL);

}

}

else

{

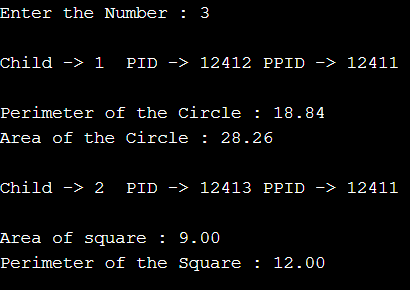
printf("\nChild -> 1 PID -> %d PPID -> %d\n", getpid(),getppid());

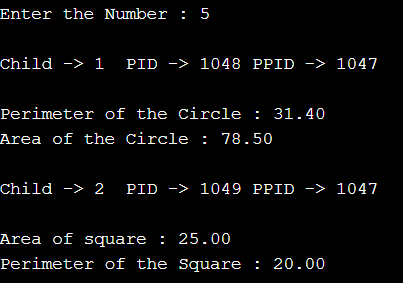
printf("\nPerimeter of the Circle : %0.2f",2\*3.14\*num);

printf("\nArea of the Circle : %0.2f\n",3.14\*num\*num);

}

}





**6. Create a parent process having two children. The first child should overwrite its address space with a process that prints “Happy new year” (happynewyear.c). The second child should overwrite its address space with another process that prints the sum of digits of a number entered by the user(sum.c).**

**[Hint: use exec family of system calls]**

**Sample output: The output should come in the following order**

**Happy new year**

**Enter the number: 123  
                     Sum of Digits: 6  
                     Parent exiting ...good bye.**

**Start.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main(int argc, char \*argv[])

{

printf("ID of the Process ( Parent ) = %d\n", getpid());

if(fork())

{

wait(NULL);

if(!fork())

{

char \*args[]={"./Happy\_New\_Year",NULL};

execvp(args[0],args);

}

else

{

wait(NULL);

printf("\nParent Process Terminating.....\n");

}

}

else

{

char \*args[]={"./Sum",NULL};

execvp(args[0],args);

}

}

**Happy\_New\_Year.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main()

{

printf("\nThis is Happy\_New\_Year which is a Child 1!");

printf("\nPID of Happy\_New\_Year.c = %d", getppid());

printf("\nHappy New Year");

printf("\nExiting Happy\_New\_Year.c......\n");

return 0;

}

**Sum.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main()

{

printf("\nThis is Sum.c which is a Child 2!");

printf("\nPID of Sum.c = %d\n", getppid());

printf("Please enter a number : ");

int num;

scanf("%d",&num);

int sum=0;

while(num>0)

{

int k=num%10;

sum+=k;

num=num/10;

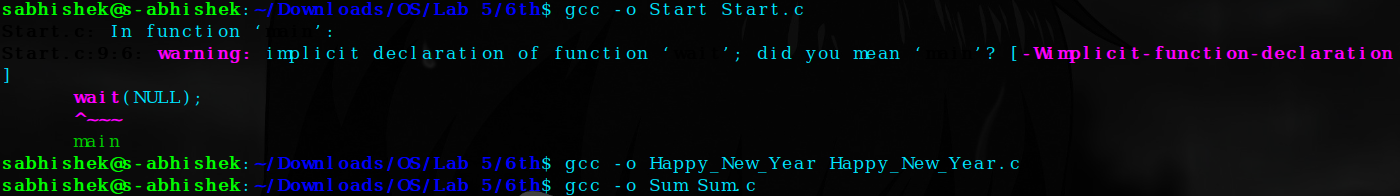
}

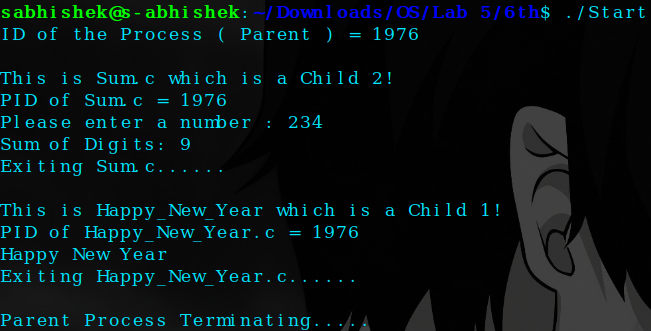
printf("Sum of Digits: %d\n", sum);

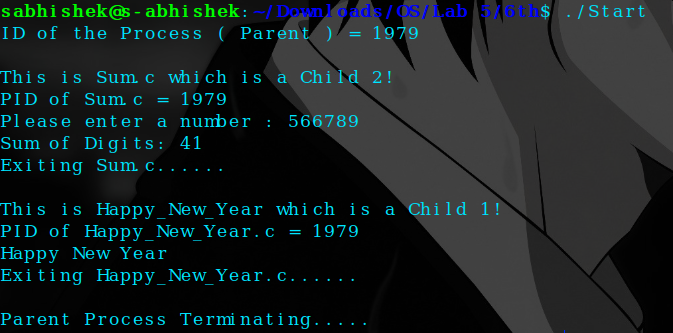
printf("Exiting Sum.c......\n");

return 0;

}







One Drive Link: [Click Me!!](https://amritauniv-my.sharepoint.com/:f:/r/personal/sabhishek_am_students_amrita_edu/Documents/S4/OS/Lab%205?csf=1&web=1&e=bhFX5L)

**Thankyou!!**