**QUES:**

**WAP (using fork() and/or exec() commands) where parent and child execute:**

**a. same program, same code**

**b. same program, different code**

**c. different programs**

**d. before terminating, the parent waits for the child to finish**

**its task**

**1) same program, same code**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

//same program,same code

int main()

{

int a;

a=fork();

if(a<0)

{

printf("child process could not be created");

exit(-1);

}

else

{

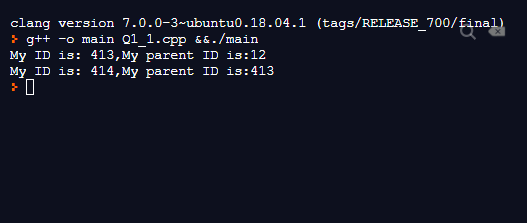
printf("My ID is: %d,My parent ID is:%d\n",getpid(),getppid());

}

return 0;

}

Output:



**2) same program, different code**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

//same program,different code

int main()

{

int a;

a=fork();

if(a<0)

{

printf("child process could not be created");

exit(-1);

}

else if(a==0)

{

printf("child : Parent process ID: %d\n",getppid());

printf("child : process ID: %d\n",getpid());

}

else{

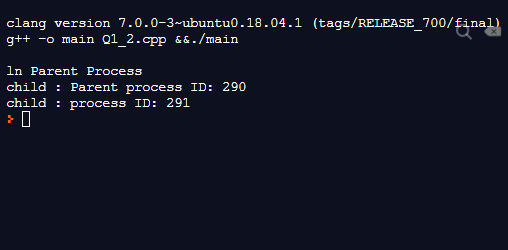
printf("\nln Parent Process\n");

}

return 0;

}

**OUTPUT:**



**3) different programs**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

//different program

int main(void)

{

int a;

a=fork();

if(a<0)

{

fprintf(stderr,"Error in fork()\n");

exit(-1);

}

else if(a>0)

{

execlp("/bin/ls","ls",NULL);

}

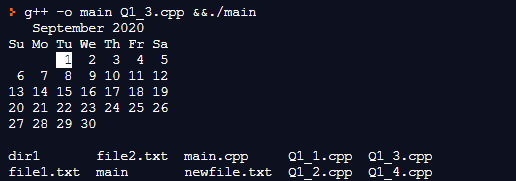
else

execlp("/usr/bin/cal","cal",NULL);

return 0;

}

**OUTPUT:**



**4) before terminating, the parent waits for the child to finish**

**its task**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

int main()

{

int a;

a=fork();

if(a<0)

{

printf("Error in the fork");

exit(-1);

}

else if(a>0){

wait(NULL);

printf("Parent:child exited\n");

}

else

{

printf("child : Parent process ID: %d\n",getppid());

printf("child : process ID: %d\n",getpid());

}

return 0;

}

**OUTPUT:**

