FORK

INPUT:

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

pid\_t q;

q=fork();

if(q<0)

{

printf("ERROR");

}

else if(q==0)//child

{

printf("child having pid %d \n",getpid());

printf("My parents pid %d \n",getppid());

}

else

{

printf("Parent having pid %d \n ",getpid());

printf("grandfather %d \n",getppid());//init process

printf("My child having pid %d \n",q);

}

printf("COMMON \n");

}

OUTPUT:

ORPAHN:

INPUT:

//orphan

//parents are dead,child is alive

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

pid\_t q;

q=fork();

if(q<0)

{

printf("ERROR");

}

else if(q==0)//child

{

sleep(2);

printf("child having pid %d \n",getpid());

printf("My parents pid %d \n",getppid());

char \*argv[2] = {"ps", NULL};

execv("/bin/ps", argv);

}

else

{

printf("Parent having pid %d \n ",getpid());

printf("My child having pid %d \n",q);

}

printf("COMMON \n");

}

OUTPUT:

The ID 1 indicating the INIT PROCESS

ZOMBIE:

INPUT:

//parent assumes ,child is in the system ,but its not there

//even though child process has terminated,entry is still there

//concept: CHILD SHOULD FINISH BEFORE PARENT

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

pid\_t q;

q=fork();

if(q<0)

{

printf("ERROR");

}

else if(q==0)//child

{

printf("child having pid %d \n",getpid());

printf("My parents pid %d \n",getppid());

char \*argv[2] = {"ps", NULL};

execv("/bin/ps", argv);

}

else

{

sleep(3);

printf("Parent having pid %d \n ",getpid());

printf("My child having pid %d \n",q);

char \*argv[2] = {"ps", NULL};

execv("/bin/ps", argv);

}

printf("COMMON \n");

}

OUTPUT:

Defunct shows the child is dead !