1. Creating child process

#include <stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main(void) {

int p;

p=fork();

if(p==0)

{

printf("%d\n",getpid());

printf("%d",getppid());

}

return 0;

}

1. Computational difference in child and parent process

#include <stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main(void) {

int p,i=5;

p=fork();

if(p==0)

{

printf("Child process%d\n",++i);

}

if(p>0)

printf("Parent Process %d\n",i);

//else

//printf("Error\n");

return 0;

}

1. Create 8 processes with minimal number of forking

#include<stdio.h>

#include<unistd.h>

int main()

{

fork();

fork();

fork();

printf("Hello\n");

}

**Orphan Process**

#include <stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main()

{

p=fork();

if(p==0)

{

printf("Child Process\n");

sleep(5);

printf("Orphan Process\n");

}

else if(p>0)

printf("Parent Process\n");

return 1;

}

**Zombie Process**

#include <stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main()

{

int p;

p=fork();

if(p==0)

printf("Child Process\n");

else if(p>0)

{

sleep(5);

printf("Parent Process\n");

printf("Zombie Process\n");

while(1)

{

};

return 1;

}

}

$ ps -A