Fork

Program A	Program B	Program C	

#include<stdio.h>
main( )
{ printf("X\n");
fork( );
printf("Y\n");
}
	main( )
	{ printf("%d ; %d\n",getpid( ),getppid( ));
	fork( );
	printf("%d , %d\n",getpid( ),getppid( ));
	sleep(2);
	}
		printf("%d\n",getpid( ));
		int p=getpid( ); fork( );
		while (getppid( )==p) ;
		printf("%d\n",getpid( ));
		printf("%d\n",getppid( ));
		sleep(2);	

Program A prints one X and two Y's. When fork system call is executed, a duplicated copy of the program is made. Both copies of the program are executed independently. Each copy is called a process.

Process Identifier (pid): When we execute a program, a process is created. Every process has got an identifier (henceforth id). It can be obtained by calling getpid( ). Different execution of the same program create different processes, hence their identifiers are different. Every process has got a father. It is the process, which created it. When we give a command to execute a program, the father is the login process. If we execute the following one line program, two numbers will be outputted. printf("%d %d\n",getpid( ),getppid( ));

On different executions of the above program, the first number will be different, but the second number will be same. It is because we are in the same window process. However, if we open a new window, then the second number will also be different.

Program B has two possible outputs. Here L is the id of the window process. A is the id of the parent process and B is the id of the process created by fork( ). The father of the process B is A.

The second and the third lines of the output may be exchanged.	output		No sleep		rare
If we remove sleep(2) then output may be different: It is because, if after the	AL	AL	AL	AL	AL
fork the parent executes first, then it will print its id, its parent (login) id, and	AL	BA	AL	BA	AL
will terminate. Now child will be adopted by a process, whose id is 1.	BA	AL	B1	AL	BA

When parent of a process dies then it is adopted by process of id 1 (It may be different on some system).

Program C The output is A A L B 1. A and B are parent and child id's respectively. B is printed 2 seconds after both A's are printed. It is because, the child keeps on waiting, till the parent is over by while (getppid( )= =p); A child can know, whether its parent is over, by comparing its current parent id by the id of the parent at the time when it was created by fork.

Program D	Program E	Program D creates a child. The parent prints X and child prints Y. The parent and child execute simultaneously. Here X and Y are mixed. Program E prints X and Y in any order. If we remove sleep(1) then child may not print any thing. Hence possible outputs are YX or X.(XY rare)

int i,p=getpid( ); fork( );
for(i=1;i<1000000;i++)
if (p==getpid( ))
printf("X");
else printf("Y");
	p=getpid( );
	fork( );
	if (p==getpid( )) printf("X");
	if (p==getppid( )) printf("Y");
	sleep(1);	

In (E) the output XY is possible only in case parent prints X and before it halts the child finds its parent's id.

Program 1	Program 2	Program 3	Program 4	Program 5

fork( );fork( );
printf("X\n");
fork( );
printf("Y\n");
fork( );
fork( );
fork( );
printf("Z\n");
	fork( );fork( );
	int p=getpid( );
	fork( );fork( );fork( );
	if (p==getpid( ))
	printf("X");
	if (p==getppid( ))
	printf("Y");
	sleep(1);
		fork( );fork( );
		int p=getpid( );
		fork( );fork( );fork( );
		if (p==getpid( )) sleep(4);
		fork( );
		if (p==getppid( )) sleep(2);
		if (p==getppid( ))printf("Y");
		sleep(1);
			int i,p=getpid( );
			for (i=1;i≤10;i++)
			fork( );
			if (p==getpid( ))
			printf("X");
			if (p==getppid( ))
			printf("Y");
			sleep(1);
				p=getpid( );
				for (i=1;i≤n;i++)
				fork( );
				if (p==getapid(r))
				printf("A");

1. In program B put p=getpid if(..)sleep(..) to get output AL,AL,B1 always (A)ALBAAL (B)ALALBA
2. How many X, Y, Z are printed by programs 1..4? (A) Remove sleep(1) from program 2. (B) Remove Line of sleep(2) from program 3. Do not use computer for them.
3. Suppose there is a function getapid(x). It returns the id of x^{th} ancestor of a process. e.g. getapid(0) is same as getpid( ). getapid(1) is same as getppid( ). getapid(2) returns the id of grandfather. How many A's will be printed by program 5 for (A) n=5 and r=2 (B) n=5 and r=3 (C) Express the number of A's in terms of n and r.
4. Create 3 process using 2 fork( )'s. No getppid( ). Minimum getpid( )'s. No loop. (A) [5 using 3][6,7/3][11/4]
5. Read n and create n processes. Every process has one child. Let printf("[%d %d]",getpid( ),getppid( )); sleep(1); is written at the end o/p for n=4 [AL],[BA],[CB],[DC],[ED]