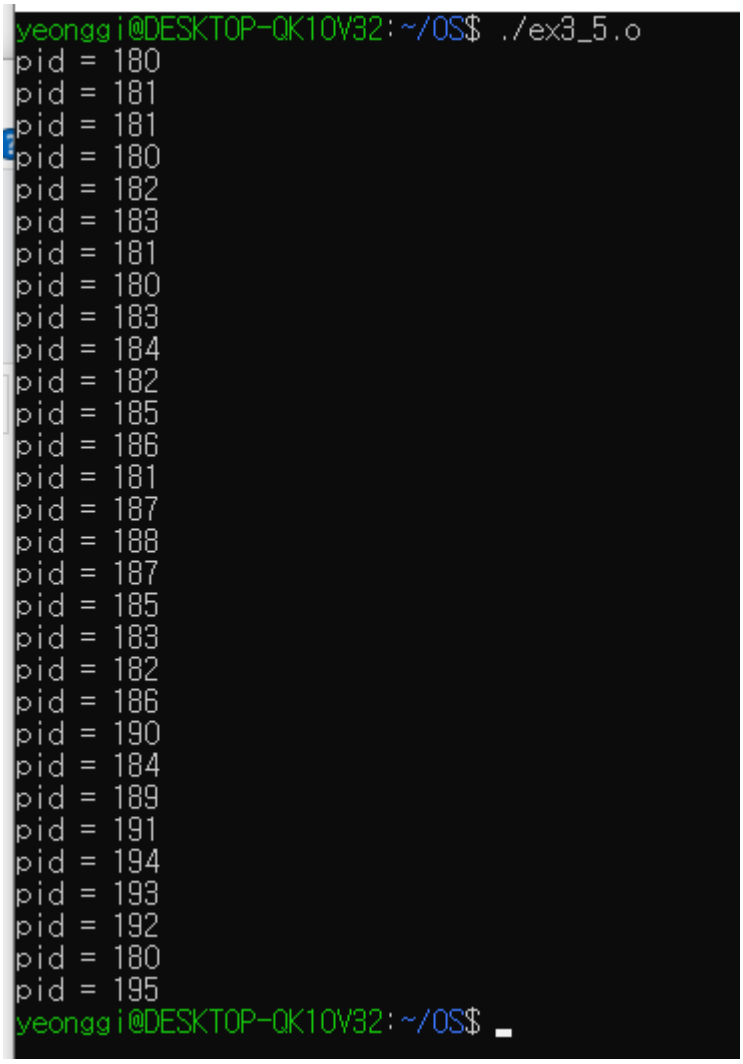
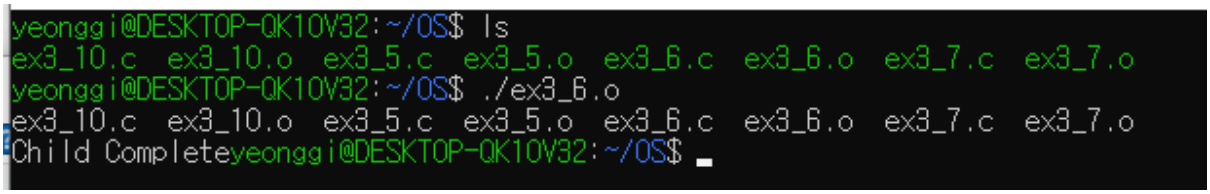
3.5 Including the initial parent process, how many processes are created?



부모까지 포함해 총 16개의 프로세스가 생성된다.

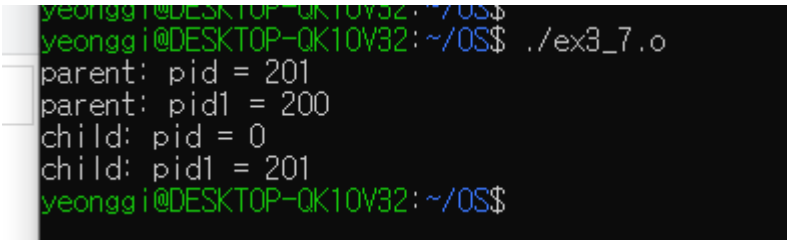
Parent 부터 시작하여 첫 fork 시 Parent와 Child1 생성. 두번째 fork 시 Parent 와 Child1 이 둘 다 복사되므로, Parent, Child1, Child2, Child3 이 생성. 세번 째 for loop 에서 다시 4 프로세스가 복사 되므로, Parent, Child1, Child2, Child3, Child4, Child5, Child6, Child7 이 생성 되고, 마지막 네번째 for loop에 진입 시 한번 더 복사 되어 Parent 와 Child 15개의 프로세스가 생성된다.

3.6 Explain the circumstances under which the line of code marked printf("LINE J") in Figure 3.32 will be reached.



execlp 함수 자체가, 시스템 콜로 자신의 메모리 공간을 새로운 프로그램으로 교체하여, 정상 실행이 된다면 뒤에 있는 printf("LINE J"); 까지 갈 수 없다. 즉 child process 는 else if (pid == 0) 까지 무사 진입을 하지만, execlp에서 ls 명령으로 덮어 써 수행하고 종료를 해버려 "LINE J"는 수행하지 않고, 만약 execlp에 문제가 생겨 실행을 하지 않는다면 printf("LINE J")구문이 실행이 될 수도 있다. 그리고 부모 process 는 child와 다른 프로세스로 else 구문으로 바로 들어가 자식 프로세스가 종료될 때 까지 기다린 후 Child Complete라는 메시지를 출력 후 프로세스가 종료된다.

3.7 Using the program in Figure 3.33, identify the values of pid at lines A, B, C, and D. (Assume that the actual pids of the parent and child are 2600 and 2603, respectively.)



A : 0

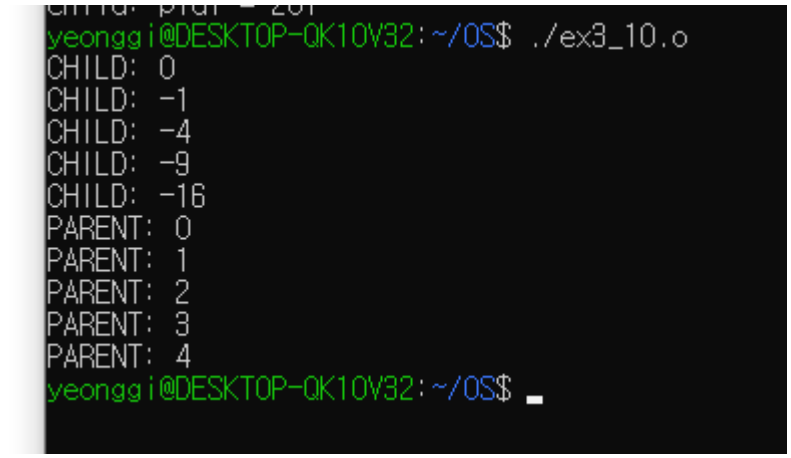
B : 2603

C : 2603

D : 2600

Parent의 pid가 2600 이고 자식의 pid 가 2603이라면, 부모부터 본다면 C 에서 fork 한 리턴값(자식 프로세스의 pid)이 pid에 들어가므로, 2603이 들어가고, D 에는 자신의 getpid인 2600 이 들어간다. 이제 자식을 보면 A에는 fork 후 child process 이므로 0이 들어가며, B 에는 자신의 process id 인 2603이 들어갈 것이다.

3.10 Using the program shown in Figure 3.34, explain what the output will be at lines X and Y.



fork는 fork 한 그 위치에서 복사가 된다. 즉 Child process 와 Parent process 둘 다 공유하지 않는 각각의 nums[5] = {0, 1, 2, 3, 4}를 가지게 된다. 그래서 실행을 하게 되면 (pid==0)에는 child가 들어가 자신의 nums[5]의 배열을 가지고 진행을 하게 되어 nums[5]배열에 수정을 하며 for loop을 돌며 0, -1, -4, -9, -16이란 값을 가지고 순차적으로 출력하게 된다. 그리고 Parent process 먼저 자식의 Process 가 끝나기를 기다리고 ,끝났다면 또 자신의 nums[5] = {0, 1, 2, 3, 4}라는 배열을 가지고 진행을 하게 되어, 차례대로 nums의 배열의 값을 출력 해 0, 1, 2, 3, 4의 값을 차례대로 출력하는 과정을 가진다.