20160074 화학공학과 고진민

[컴퓨터 SW시스템 개론](http://lms.postech.ac.kr/Main.do?cmd=viewCourseMain&mainDTO.courseId=20180920140072901661213&gubun=class_learner) HW5

1. (8.14)

총 **4번** 출력된다.

Try 함수는 fork() 함수를 수행해 pid가 0이 아니면 fork()를 한 번 더 수행한 후 Example을 출력하고 프로그램을 끝낸다. Main 함수를 보면 먼저 try 함수를 호출한다. Try 함수의 if문 내에서 fork()가 호출되면, parent에 대한 child process가 생긴다. 이를 child 1이라 하자. 단 child 1은 fork()가 0을 반환하므로 if문 내부를 진입하지 못하고 return 한다. Parent process는 if문 내부로 들어가 fork()를 한 번 더 호출해 추가 child process를 생성(child 2)하고 이 child process는 if문 내부에 있는 상황이다. 그러면 parent process와 child 2는 각각 Example을 출력하고 exit 함수에 의해 종료된다. Child 1은 return 되어 main 함수로 돌아와서 fork()를 호출해 child 1의 child인 child 3을 생성한다. 이 child 1과 child 3이 각각 Example을 출력한 후 exit 함수에 의해 종료된다. 이렇게 총 **4번**의 Example line이 출력된다.

1. (8.18)



위 도표를 참고하면, 가능한 출력은 **A, C, E**이다. B는 2가 0이나 1보다 먼저 출력될 수 없어서 불가능하며 D 역시 2가 하나는 1이 나온 뒤에 출력되었으나 다른 하나가 0 이전에 출력될 수 없으므로 불가능하다.

1. (8.19)

총 **8회** 출력된다.

for문은 총 3번 돌게 된다. n = 6이고 5, 3, 1에서 각각 fork가 호출된다. fork()는 매 수행마다 child를 생성하므로, 호출이 될 때마다 전체 process의 수가 2배가 된다고 생각하면 for문 종료 후 전체 process의 수는 8개가 된다. 이들이 각각 hello를 출력하므로 총 8회 출력된다.

1. (8.21)

**pqr, prq**가 가능하다. 처음에 p가 출력된 후 fork를 수행해 parent면 q를, child는 r을 출력하게 되는데 child의 경우 waitpid가 있으므로 child의 child가 전부 종료되어야 호출된다. 그러나 현재 새롭게 생성된 child의 child는 없으므로 q나 r 중 무엇이 먼저 출력될지 알 수 없으므로 둘 모두 가능하다.

1. (10.6)

아래 상황은 foo.txt, bar.txt, baz.txt가 모두 존재하는 경우에 대한 것이다. Open 함수는 descriptor를 return할 때 열리지 않은 descriptor 중 가장 작은 descriptor를 반환한다. 처음 fd1으로 foo.txt를 open할 때 3이 할당되고(0, 1, 2는 각각 stdin, stdout, stderr이므로) fd2로 bar.txt를 open할 때 4를 할당했다가 fd2가 close하면서 4가 반환된다. 이후 baz.txt를 fd2로 여는 과정에서 다시 4가 할당되고, 이 때문에 fd2를 출력하면 **4**가 출력된다.

1. (9.11)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

B.

|  |  |
| --- | --- |
| Parameter | Value |
| VPN | 0x9 |
| TLB index | 0x0 |
| TLB tag | 0x2 |
| TLB hit? | N |
| Page fault? | N |
| PPN | 0x17 |

C.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

D.

|  |  |
| --- | --- |
| Parameter | Value |
| Byte offset | 0x0 |
| Cache index | 0xf |
| Cache tag | 0x17 |
| Cache hit? | N |
| Cache byte returned | - |

1. (9.13)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

B.

|  |  |
| --- | --- |
| Parameter | Value |
| VPN | 0x1 |
| TLB index | 0x1 |
| TLB tag | 0x0 |
| TLB hit? | N |
| Page fault? | Y |
| PPN | - |

C.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | - | - | - | - | - | - |

D.

|  |  |
| --- | --- |
| Parameter | Value |
| Byte offset | - |
| Cache index | - |
| Cache tag | - |
| Cache hit? | - |
| Cache byte returned | - |