**CSED211 Homework 5**

20210643 Kim Hyun-June

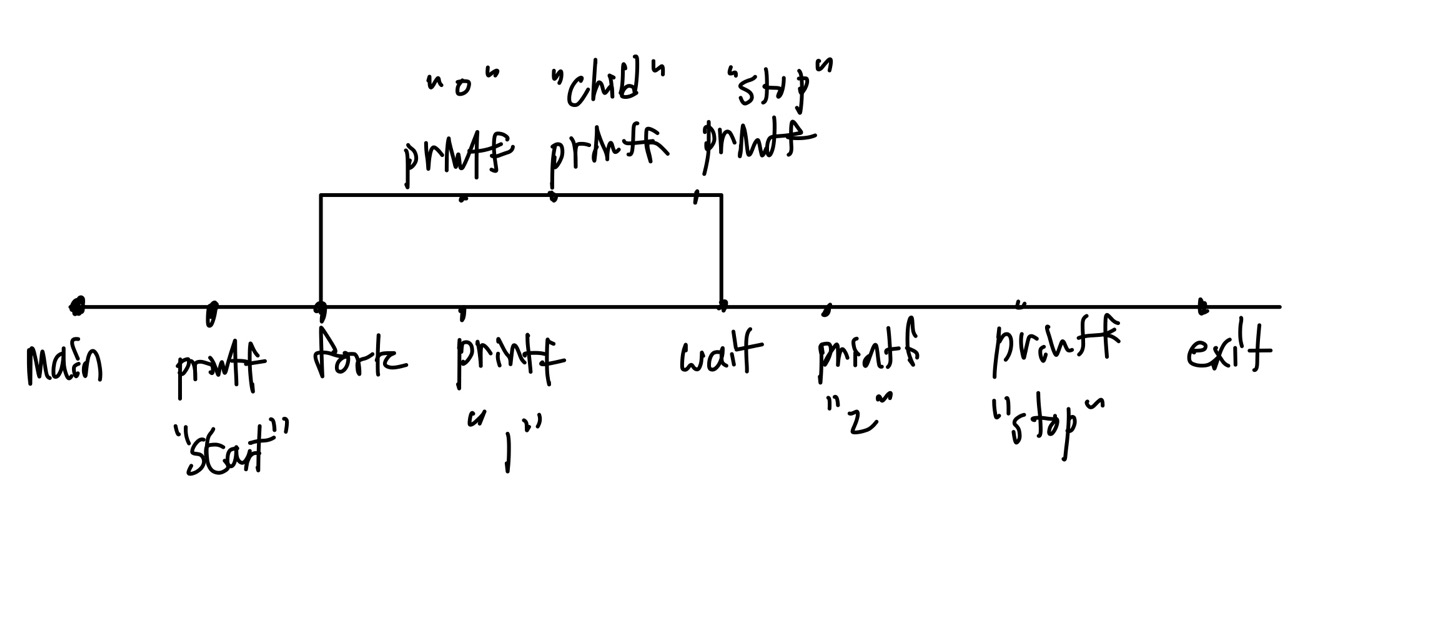
**1. Exercise 8.16 on page 826.**

main()에서 fork()이 실행되면서 자식 프로세스가 생겨난다. 자식 프로세스의 counter는 따로 존재할 것인데, 그 값이 1 증가되어 2가 된다. 이후 wait(NULL)에 의해 자식 프로세스가 종료될 때까지 부모 프로세스가 기다리고, 부모 프로세스는 own counter를 갖고 있는데 그 값이 ++된 이후 출력되므로 counter = 2 가 출력될 것이다.

답: counter = 2

**2. Exercise 8.17 on page 827**

Process graph를 그려보면 아래와 같다.



먼저 Start를 print한 다음, Fork()함수를 호출하면서 프로세스가 생기고, 각 프로세스에서 1과 0, Child, Stop을 출력하고, wait 이후 다시 하나의 프로세스에서 2와 stop을 출력한다. 따라서 앞 부분 Start와 뒷부분 2 Start는 항상 같고 그 사이에서 두 프로세스 부분에서 여러 경우의 수가 생길 것이다. 경우의 수를 모두 고려해보면 아래와 같은 것들이 가능하다.

답(출력되는 순서대로):

Start, 1, 0, Child, Stop, 2, Stop

Start, 0, Child, Stop, 1, 2, Stop

Start, 0, Child, 1, Stop, 2, Stop

Start, 0, 1, Child, Stop, 2, Stop

**3. Exercise 8.19 on page 828 with n = 7**

n=7 이므로 for문은 n=7에서 시작하므로 i=6, i=4, i=2, i=0 순으로 Fork() 함수를 호출하는 부분이 4번 돌아가게 된다. 따라서, 자식 프로세스가 2^4=16개 만들어지고, 각 프로세스마다 hello를 한 번씩 print할 것이므로, 총 16줄의 hello가 print될 것이다.

답: 16 lines

**4. Exercise 8.21 on page 828**

먼저, p는 맨 앞에 출력된다. 그 다음에 q가 먼저 출력되고 그 다음 r이 출력되는 경우와 그 반대 경우가 있을 것이다. r이 출력된 이후 다른 프로세스를 대기하다가 q가 출력 이후 대기 상태가 종료되고 프로세스가 종료되는 경우가 r 다음으로 q가 출력이 가능한 경우일 것이다. 결론적으로 pqr, prq의 두 가지가 가능하다.

답: pqr, prq

**5. Exercise 9.11 on page 912**

**A. Virtual address format**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

Virtual address 0x027c=1001111100(2) 이므로 위와 같다.

**B. Address translation**

|  |  |
| --- | --- |
| Parameter | Value |
| VPN | 0x9 |
| TLB index | 0x1 |
| TLB tag | 0x2 |
| TLB hit? (Y/N) | N |
| Page fault? (Y/N) | N |
| PPN | 0x17 |

Virtual Page Number가 6~11 부분이므로, VPN은 00001001(2)=0x09 이다.

TLB index와 TLB tag도 찾으면 각각 0x1, 0x2이다.

표에서 tag를 찾으면 valid set이 없기 때문에 TLB hit는 일어나지 않을 것이다.

Page table에서 Page fault 여부를 확인한 결과 fault가 일어나지 않는다.

PPN 또한 page table에서 찾으면 0x17이다.

**C. Physical address format**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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. Physical memory reference**

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

Cache table에서 해당 index에 일치하는 tag가 존재하지 않으므로 miss가 일어난다. 따라서 cache return 값도 없다.

**6. Exercise 9.14 on page 914**

**7. Exercise 9.15 on page 915**

**8. Exercise 9.16 on page 915**

