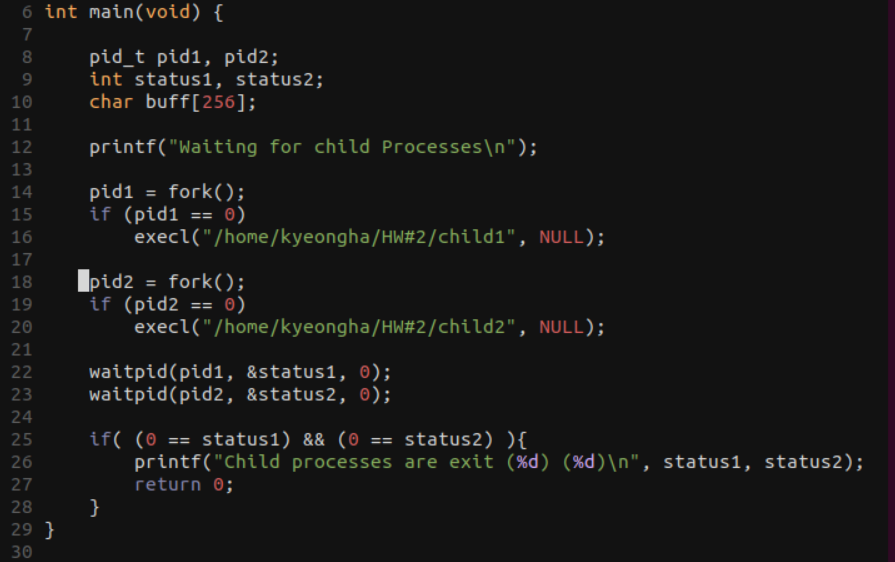
**2018008304 컴퓨터소프트웨어학부 박경하**

**운영 체제 HW#2**

**제출 일자: 2021/03/30**

1. **과제 A**
2. **부모 프로세스, 자식 프로세스1, 자식 프로세스2 코드 설명**

1) 부모 프로세스



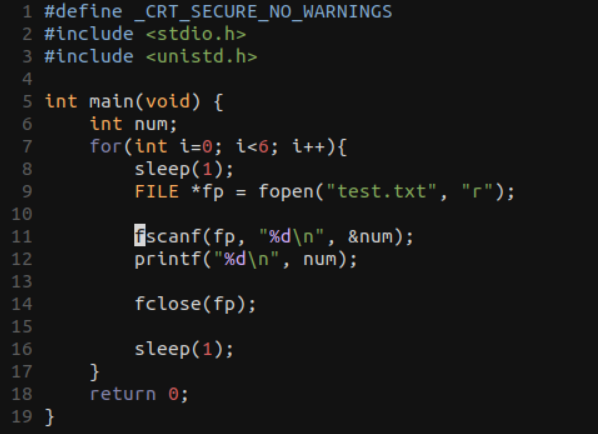
부모 프로세스에서는 먼저 자식 프로세스를 기다린다는 문구를 출력해준 후, pid를 2개 만들어 각각 fork-exec를 진행하였습니다. pid1는 자식 프로세스1으로, pid2는 자식 프로세스2로 만들어지도록 하였으며 동작을 모두 마친 뒤, 즉 자식 프로세스가 모두 종료되었음을 알기 위해 최하단에 waitpid를 이용하여 pid1, pid2의 status값이 0이 되면 모든 자식 프로세스가 종료된 것으로 감지하고 부모 프로세스를 종료하도록 설계하였습니다.

2) 자식 프로세스1



자식 프로세스1은 출력결과를 보았을 때 11까지 증가하여 for문의 i값을 11까지로 맞췄습니다. 그리고 1초 간격을 구현하기 위해 sleep을 이용하였고, 후에는 파일입출력을 통해 i값을 넣어 해당 조건을 구현했습니다.

3) 자식 프로세스2

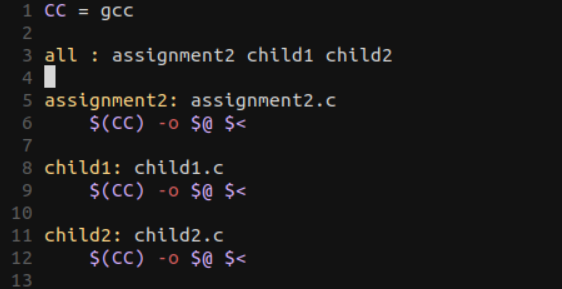


자식 프로세스2는 출력결과의 타임을 따르기 위해 sleep을 맨 앞과 맨 뒤로 나누어 넣어놨습니다. 마찬가지로 파일 입출력을 통해 파일의 첫 줄을 읽어 num에 저장한 후 출력하는 형태로 구현했으며, 모든 동작을 마친 뒤에는 0값을 return하도록 했습니다.

1. **구현한 fork-exec 구조 설명**

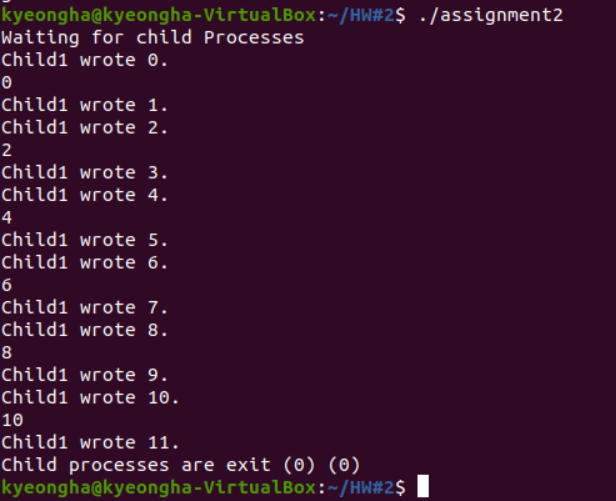
fork()로 구현한 자식 프로세스는 부모의 코드를 그대로 따를 수 밖에 없기 때문에 exec를 통해 child1, child2로 넘어가도록 하였습니다. 따라서 부모 프로세스에서 exec를 호출하게 되면 새로 작성된 child1.c과 child2.c의 코드로 대체하게 되며, 작성된 자식 프로세스에서 for문을 통해 모든 작업을 마친 뒤에 0값을 리턴하여 자식 프로세스가 종료됨을 알립니다.

1. **컴파일 방법 설명 (Makefile 등)**



Makefile은 3가지의 실행파일을 만들기 위해 all을 이용하였으며 그 뒤로는 1개의 컴파일할 때와 동일하게 작성하였습니다.

1. **실행 결과**

****

출력결과는 영상에서 본 것과 같으며, 마지막 괄호 안에는 status 값이 나오도록 만들었습니다.

1. **과제 B**
2. **IPC 내용**

IPC란 producer와 consumer간에 커뮤니케이션 하는 것을 의미하며, 여러가지 각도에서 다양한 의미와 구현을 살펴볼 수 있다.

**1. 메카니즘을 어떻게 구현할 것인지에 따라**

1) Shared memory

프로세스 간의 공유할 수 있는 공유메모리 영역을 지정하는 방법으로, 프로듀서는 공유메모리 영역을 write하고 컨슈머는 read하는 형태를 갖는다.

2) Message passing

공유메모리를 전제하는 것이 아니라 네트워크 채널을 통해서 프로듀서가 메세지를 보내고, 컨슈머는 받는 send와 recive 형태를 갖는다.

**2. 커뮤니케이션 Link가 어떻게 구현되는 지에 따라**

1) Logical link

물리적인 연결방법과 무관하게 프로그래밍 관점에서 어떤 커뮤니케이션 링크를 쓰는 것을 구분한다. 물리적으로는 shared를 이용하지만 프로그래밍 관점에서는 shared의 write/read가 아닌 Message의 send/receive를 사용하여 구현하는 경우를 예로 들 수 있다.

2) Physical link

물리적으로 실제 커뮤니케이션이 구현되는 링크를 말하며 Shared/Message 중 어떤 것을 쓸 것인가를 구분한다.

**3. 커뮤니케이션 타입에 따라**

1) Direct Communication

프로세스들이 직접적으로 메세지를 주고 받는 경우를 의미한다. 일반적으로 send(P, message)를 쓰고, P는 메세지를 받는 데스티네이션 프로세스가 되며, P 입장에서는 receiver(Q, message)를 쓰게 되고 Q라는 소스 프로세스를 적는다.

2) Indirect Communication

메세지를 주고받는 상대를 지칭하지 않는 대신에 제3의 영역을 지정하는 메일박스가 존재한다. send함수를 호출할 때 P를 지칭하는 것이 아니라 A라는 메일박스를 지칭하게 되어 있다.

**4. 커뮤니케이션을 진행하는 방식에 따라**

1) Synchronous

메세지를 주고 받을 때 그 프로세스가 blocking을 받는 경우를 의미한다.

1) Asynchronous

메세지를 주고 받을 때 그 프로세스가 blocking을 받지 않는 경우를 의미한다.