**2017029589 컴퓨터소프트웨어학부 류지범**

**운영 체제 HW#2**

**제출 일자: 2021/03/23(화)**

1. **과제 A**
2. **부모 프로세스, 자식 프로세스1, 자식 프로세스2 코드 설명**
   1. 텍스트이(가) 표시된 사진

      자동 생성된 설명부모 프로세스
      1. Pid1, pid2는 자식 프로세서 2개를 위한 변수이고, status1, status2는 각각 자식 프로세스에 대한 waitpid를 위한 인자이고, 종료 상태를 반환한다.(정상적이라면 0 반환)
      2. 10 line에서 fork()를 통해 첫번째 자식 프로세스를 복제해준다. 이후 execl함수를 통해 child1을 실행시킨다
      3. 15 line에서 fork()를 통해 두번째 자식 프로세스를 복제해준다. 이 때 바로 child2를 실행하면, 첫번째 자식 프로세스가 파일에 숫자를 기록 하는 시간과, 두번째 자식 프로세스가 파일을 읽는 시간이 겹치므로, usleep(500000)을 통해 0.5초 기다려준다. 이후 execl을 통해 child2를 실행시킨다.
      4. 21, 22 lines에 waitpid를 통해 두 개의 자식이 종료될 때까지 기다렸다가 정상적으로 종료되면 종료 문구를 출력하고 부모 프로세스도 종료한다.
   2. 텍스트이(가) 표시된 사진

      자동 생성된 설명자식 프로세스1
      1. 0~11까지의 숫자를 기록한다.
      2. 7 line에서 Test.txt라는 파일을 연다.(없으면 만들고, 있으면 덮어씌움)
      3. 8 line에서 파일에 숫자와 개행문자를 쓰고, 9 line에서 화면에 해당 내용을 출력하고, 10 line에서 file을 닫는다.
      4. 11 line에서 sleep(1)을 통해 1초간 쉬어준다.
   3. **텍스트이(가) 표시된 사진

      자동 생성된 설명**자식 프로세스2
      1. 출력할 문자를 str에 저장하기 위해 6 line에서 char str[5]를 선언
      2. 반복문을 통해 6번의 과정을 반복하고, 파일을 읽기모드로 연다.
      3. 파일의 첫줄을 읽어오고, “child2 : “ 형식으로 출력한다.
      4. Sleep(2)를 통해 2초간 쉬어준다.
      5. 14 line에서 파일을 닫아주고, 프로세스를 종료한다.
3. **구현한 fork-exec 구조 설명**
   1. 두 번의 fork를 통해 두 개의 자식 프로세스를 만든다.
   2. 각각의 fork된 자식 프로세스에서 execl를 통해 각자 파일을 읽고 쓰는 프로그램을 실행시킨다.
   3. Execl를 통해 프로그램을 실행시키면, 새로운 프로세스가 생기는 것은 아니고, 복제된 자식 프로세스에 덮어 쓰여지게 된다. Pid도 그대로 적용된다.
   4. 부모 프로세스는 waitpid를 통해 이 자식 프로세스들을 기다리다가 둘 다 종료되면 함께 종료한다.
4. **텍스트이(가) 표시된 사진

   자동 생성된 설명컴파일 방법 설명 (Makefile 등)**
   1. 컴파일은 Makefile을 통해 진행했고, 위의 사진의 코드와 같다.
   2. 세 개의 코드를 Makefile을 통해 한 번에 컴파일하였다.
5. **실행 결과텍스트이(가) 표시된 사진

   자동 생성된 설명**
6. **과제 B**
7. **IPC 내용**

IPC(Interprocess Communication)란 일반적인 관점에서 producer와 consumer간에 커뮤니케이션하는 방법이다. 여러가지 각도에서 의미와 구현 방법을 살펴볼 수 있는데, 우선 메커니즘으로는 shared memory communication과 message passing communication을 사용한다. Shared memory communication은 프로세스간의 공유할 수 있는 공유 메모리 영역을 지정하고 producer는 공유 메모리 영역에 write하고, consumer는 공유 메모리 영역을 read하는, 프로그래밍 관점에서는 write와 read의 형태로 구현할 수 있다. Message passing communication은 공유 메모리를 전제하는 것이 아니라, 네트워크 채널을 통해서 producer가 메시지를 보내고, consumer는 메시지를 받는 것이다.

다음으로 물리적인 연결 방법과, 논리적인 연결방법으로도 구분할 수 있는데, logical link는 ‘물리적인 연결 방법과 무관하게 프로그래밍 관점에서 어떤 communication link를 쓰는가’이고, physical link는 물리적으로 실제 communication이 구현되는 링크이다.

커뮤니케이션 타입으로는, Direct & indirect communication과 Synchronous & asynchronous communication이 있다. Direct communication은 프로세스들이 직접적으로 메시지를 주고받는 타입이고, 메시지를 받을 대상과, 보낸 대상이 명시된다. Indirect communication은 메시지를 주고받는 상대방을 지칭하지 않고 mailbox를 통해 메시지를 넣고 꺼내온다. Synchronous는 메시지를 주고 받을 때 프로세스가 blocking을 당하는 경우이고, asynchronous는 non-blocking한 상태이다.