**<교수님이 작성하신 문제입니다>**

**- 저작권이 문제가 될 것 같아 면접이 끝나면 바로 내리도록 하겠습니다.**

**목표**

본 과제에서 thread와 함께 fork, exec를 사용하여 OS에서 프로세스를 어떻게 운용하는지와 프로세스간 통신에 대한 이해를 높인다.

**요구기능**

이 과제에서는 datasplit.c의 기능을 확장한다. 이전 과제에서는 datasplit.c의 thread는 단순히 소문자를 대문자로 변경한 후 receiver에게 전달하는 기능을 하였지만, 이 과제에서는 thread가 사용자가 주어진 명령을 수행하도록 구현한다.

사용자는 다음과 같이 명령을 내린다.

$ cat largefile.txt | ./datasplit -n 4 "grep abc"

이렇게 하면 datasplit은 네 개의 thread가 각각 grep이라는 명령을 수행하여 그 결과를 receiver thread에 전달하게 된다. 이때 grep abc는 입력된 문장에서 abc라는 단어가 있는 것만 골라내는 작업을 한다. 쌍따옴표 안의 수행 명령은 입력을 standard in으로 받고 출력은 standard out으로 하는 어떤 프로그램이라도 올 수가 있다.

아래와 같이 여러 명령을 연결해서 입력으로 줄 수도 있다.

$ cat largefile.txt | ./datasplit -n 4 "grep abc -> grep 123"

이번에는, 한 줄에 abc라는 단어가 있고 123도 있는 줄만을 골라내는 일을 한다. 두개의 grep명령은 "->"라는 기호로 연결되어 있다. 첫번째 grep의 결과가 두번째 grep의 입력으로 들어가는 것이다. 이와 같이, 임의의 개수의 명령을 ->로 연결할 수 있다. "->"의 앞뒤에 스페이스가 있거나 없거나 동일하게 동작하여야 한다.

$ cat largefile.txt | ./datasplit -n 4 "grep abc -> grep 123 -> awk '{print $1}' -> sort -u -> wc -l"

위와 같이 하면 abc와 123을 가지고 있는 라인만을 골라서 첫번째 단어만을 추출한 후 그것들을 정렬하고 중복을 없애서 그 줄수를 숫자로 출력하는 작업을 하게 된다. 이 예에서는 -n 4이므로 최종으로는 화면에 숫자 네개가 찍힌다.

**내부구조**

쌍따옴표내의 각각의 명령은 fork와 exec를 사용해서 생성되는 개별 process이다. 각각의 process들은 standard in으로부터 입력이 들어오기를 기대하고 결과는 standard out으로 출력하게 되어 있다.

각 thread는 생성이 된 후, 필요한 개수의 pipe()를 호출하여 pipe descriptor들을 미리 준비한다. 명령이 한개이면 thread에서 fork한 그 하나의 process에게 데이타를 전달하기 위한 pipe 한쌍, 그리고 명령의 수행 결과를 다시 읽어올 pipe 한쌍, 이렇게 두쌍이 필요하게 된다.

위의 첫번째 명령 예를 사용하면, 데이타는 다음과 같이 흐르게 된다.

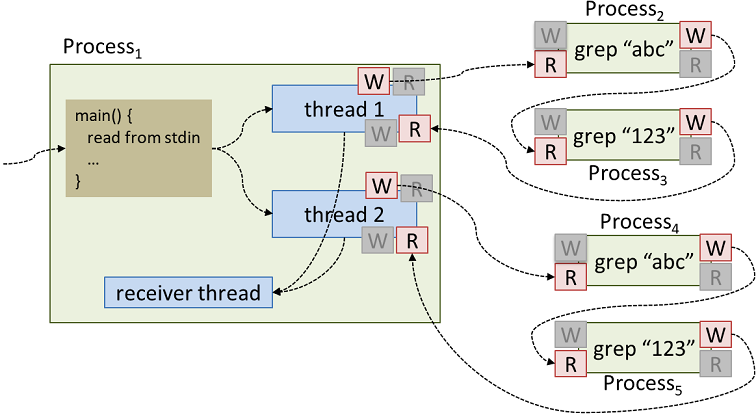
    thread1 ----> "grep abc" process ----> thread1

Thread1은 main으로부터 전달받은 한줄의 데이터를 첫번째 pipe의 write end에 write를 한다. 중간의 grep은 지시받은 작업을 수행하고 결과를 내보낸다. 그러면 다시 thread1은 두번째 pipe의 read end로 부터 읽는다. 그런 후 한줄이 다 모아지면 이전처럼 receiver thread로 전달하기만 하면 된다.

중간의 grep process는 fork가 된 후, 그리고 exec하기 전, 불필요한 pipe는 반드시 close를 하고 dup2를 써서 standard in과 standard out을 pipe로 변경하여야 한다.  dup2시스템콜은 지정한 descriptor를 강제로 닫고 주어진 descriptor를 그 descriptor번호로 복제를 하는 기능을 가진다. 따라서, pipe로 생성한 read end를 그 process의 0번으로 dup2하면 그 process는 0번(즉, standard in)으로부터 데이터를 읽어들인다고 가정하고 read를 수행하지만 실제로는 pipe의 read end로부터 데이터를 읽게 되는 것이다. 이와 유사하게, write end를 그 process의 1번으로 dup2를 하면 grep의 결과는 내부적으로는 standard out으로 write되지만 pipe를 타고 thread 1으로 전달되게 된다. 리눅스에서 man dup2를 하여 세부사항을 학습한 후 사용한다.

요구기능에서의 두번째 예, 즉 grep이 2개 있는 경우는 추가로 두 grep process들을 연결할 pipe가 하나 더 필요하게 된다. 따라서 pipe시스템콜을 세번 호출하여 세 pipe를 준비하여야 한다.

아래의 그림은 "-n 2"이며 두개의 grep을 연결한 구조를 보여주고 있다. 아래와 같이 먼저 데이터가 흘러갈 길을 다 완성한 후 main에서는 read를 해서 데이터를 받고 write로 thread에 데이타를 흘려줘야 함. **(주의)** 매번 한줄 읽을 때 마다 이렇게 fork하고 pipe연결하라는 의미가 아님.



Thread에서 명령들의 결과를 다시 읽어들일 때에는 read를 하는데 이때 pipe의 read end는 non-blocking모드이어야 한다. Blocking모드이면 thread가 read에서 걸려있어서 main이 thread에게 전달해 주는 데이터를 처리할 수 없게 된다. Non-blocking설정은 다음과 같이 한다.

    fcntl(pfd[0], F\_SETFL, O\_NONBLOCK);

**제출물**

datasplit.c 파일만 제출한다.

**주의사항**

- 마감 후 제출은 허용되지 않음. 이메일로 제출하지 말 것.

- 코드를 카피하는 것에 대한 정책은 이전 과제와 동일함.

- 이전 과제의 주의사항이 모두 적용됨.

- 현재 과제 배점은 임시이며 조정될 수 있음.

