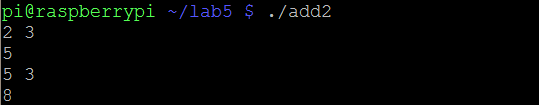
**LAB5 20113315 이형준.**

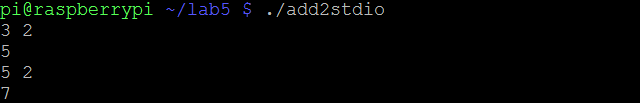
1. [강의노트](http://advsys.cs.kookmin.ac.kr/home/Lecture7-Interprocesscommunications.pdf?attredirects=0&attachauth=ANoY7cpx0CYf3-X8U-oZ2Rm1B6IZNQoOnpba9wbwBChuS9h5DCSNbmsN9MjKksSSjP0g18-JFNtoYOdqHDqzW2vPhYwhEiiNBvwk0cLKBmgruHWGHWBB_mgwEuriqLt60L9WcZL4tYaBG_or2nUdIcDxFG-p8tXdH1CAtBx5IMkBTZnfaG6Yb2B6aDcgqpbm4UOqmyObBaD62D3DMetHM4c5_KIzpEPHvBBwokURA5aBMPXKrIDkXZP777a46tGCyhfwQCowtrba) p26-29의 add2.c 프로그램을 실행시키고 입출력을 관찰하시오.



두 수를 STDIN으로 입력받아, 일련의 계산과정(덧셈) 을 거쳐 STDOUT 로 출력하는 프로그램 입니다. Read, write를 사용하여 unbuffered IO입니다.

따라서 바로 바로 출력하는 프로그램입니다.

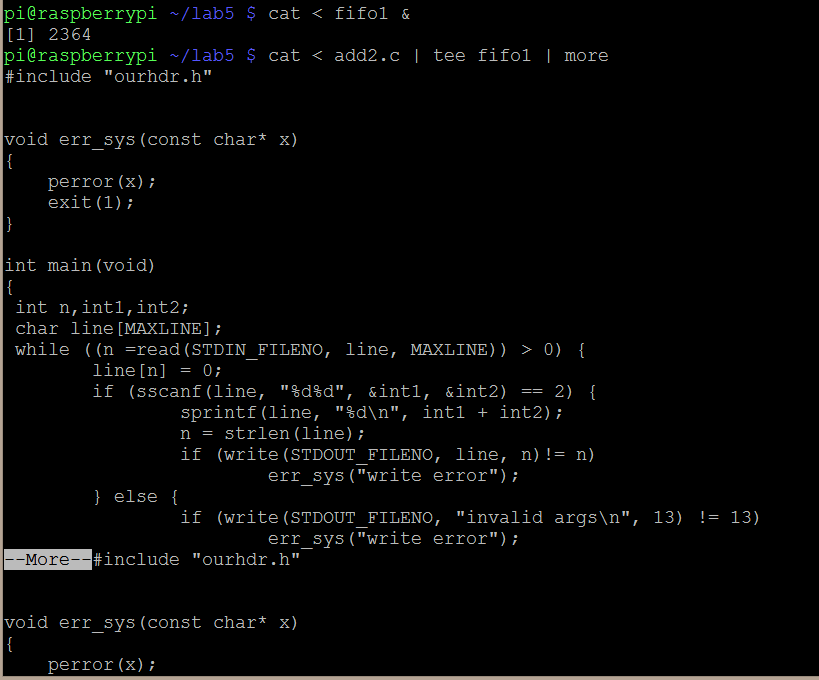
1. add2 를 p30 의 내용으로 바꾸고 실행한 뒤 입출력을 관찰 하시오.



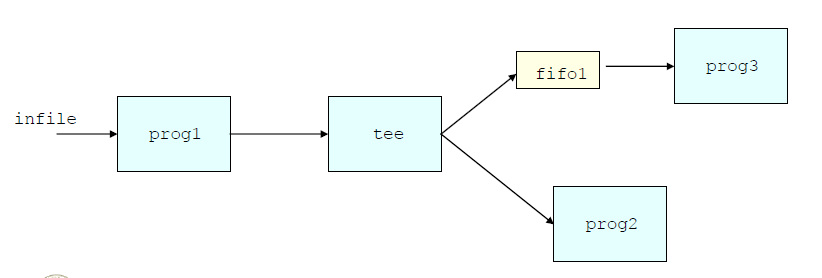
위의(1번) add2 와 같이 STDIN STDOUT은 같으나, fgets 를 사용하여

Buffered IO입니다. 터미널이 붙어있어서 line 을 기준으로 출력을 합니다.

1. p35 의 내용대로 실행하고 관찰하시오



처음 mkfifo 를 이용해서 fifo1을 만들었습니다.



구조는 위의 사진과 유사합니다.

Infile -> add2.c

Prog1 -> cat

Prog 2 -> more

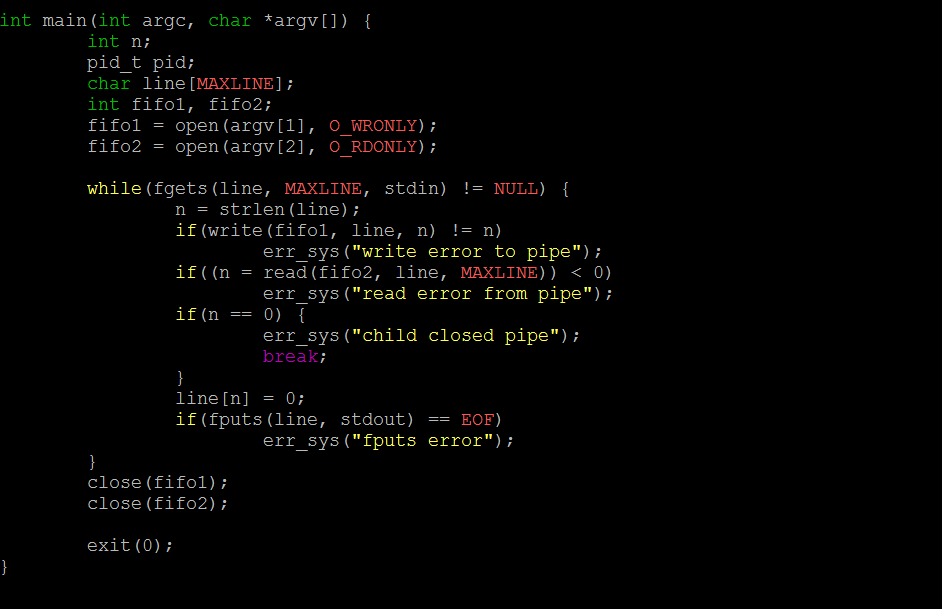
Prog 3 -> cat

을 사용하였습니다.

Cat의 입력으로 add2.c를 받았고, tee로 인해서 fifo1과 more로 출력을 동시에 하였습니다. 파이프를 동시에 두개를 사용할 수 없어서, 나머지 하나는 named pipe를 이용하였습니다. 출력 화면을 통해서 Add2.c의 코드가 두번 나오는 것을 알 수 있습니다.

1. p28 을 다음과 같이 개조하여 dummy.c 로 만들고 이 dummy 뒤에 add2 를 coprocess 로 붙여서 실행하시오.

우선 dummy.c의 소스코드입니다.

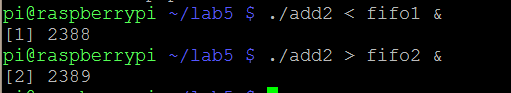


dummy.c 는 argv로 두개의 파라미터를 받습니다. 그리고 stdin으로 키보드에서 스트링을 읽어옵니다. 두 파라미터는 이름이고 첫번째 파라미터에 출력을 하고 두번째 파라미터에서 읽습니다.

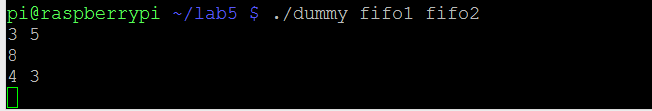
즉, 첫번째 파라미터로 STDIN을 통해 얻은 스트링을 쓰고, 나중에 읽어와서 STDOUT으로 출력하는 프로그램이 dummy입니다.



그리고 mkfifo명령어를 통해서 fifo1 fifo2를 만들었습니다.



Add2가 입력은 fifo1로부터, 출력은 fifo2로 합니다. 백그라운드로 돌려줍니다.



그리고 dummy를 fifo1는 출력, fifo2는 입력으로 파라미터를 설정해 주면 정상적으로 coprocess 가 형성됩니다.