|  |
| --- |
| #include <fcntl.h>  #include <stdio.h>  #include <errno.h>  #define MSGSIZ 63  char \*fifo = "fifo";  main (int argc, char \*\*argv)  {  int fd;  char msgbuf[MSGSIZ+1];  if (mkfifo(fifo, 0666) == -1)  {  if (errno != EEXIST)  perror ("receiver: mkfifo");  }  if ((fd = open(fifo, O\_RDWR)) < 0)  perror ("fifo open failed");  for(;;)  {  if (read(fd, msgbuf, MSGSIZ+1) <0)  perror ("message read failed");  printf ("message received:%s\n", msgbuf);  }  } |
|  |

|  |
| --- |
| #include <fcntl.h>  #include <stdio.h>  #include <errno.h>  #define MSGSIZ 63  char \*fifo = "fifo";  main (int argc, char \*\*argv)  {  int fd;  char msgbuf[MSGSIZ+1];  if (mkfifo(fifo, 0666) == -1)  {  if (errno != EEXIST)  perror ("receiver: mkfifo");  }  if ((fd = open(fifo, O\_RDONLY)) < 0)  perror ("fifo open failed");  for(;;)  {  if (read(fd, msgbuf, MSGSIZ+1) <0)  perror ("message read failed");  printf ("message received:%s\n", msgbuf);  }  } |
|  |

1) O\_RDWR 을 O\_RDONLY로 바꿨을 때 결과.

2-1) read 반환 값이 <=0 일 때까지 에러처리

|  |
| --- |
| #include <fcntl.h>  #include <stdio.h>  #include <errno.h>  #define MSGSIZ 63  char \*fifo = "fifo";  main (int argc, char \*\*argv)  {  int fd;  char msgbuf[MSGSIZ+1];  if (mkfifo(fifo, 0666) == -1)  {  if (errno != EEXIST)  perror ("receiver: mkfifo");  }  if ((fd = open(fifo, O\_RDWR)) < 0)  perror ("fifo open failed");  for(;;)  {  if (read(fd, msgbuf, MSGSIZ+1) <= 0)  perror ("message read failed");  printf ("message received:%s\n", msgbuf);  }  } |
|  |

2-2) O\_RDWR 을 O\_RDONLY로 바꾸고 read 반환 값이 <=0 일 때까지 에러처리

|  |
| --- |
| #include <fcntl.h>  #include <stdio.h>  #include <errno.h>  #define MSGSIZ 63  char \*fifo = "fifo";  main (int argc, char \*\*argv)  {  int fd;  char msgbuf[MSGSIZ+1];  if (mkfifo(fifo, 0666) == -1)  {  if (errno != EEXIST)  perror ("receiver: mkfifo");  }  if ((fd = open(fifo, O\_RDONLY)) < 0)  perror ("fifo open failed");  for(;;)  {  if (read(fd, msgbuf, MSGSIZ+1) <= 0)  perror ("message read failed");  printf ("message received:%s\n", msgbuf);  }  } |
|  |

Receiver 에서 RDWR 로 fifo 를 열어야 하는 이유: fifo 에서 read를 실행할 때, WR권한이 있는 프로세스가 있는 상태에서 \0을 만나면 WR권한이 있는 프로세스에서 write를 할 때까지 대기상태로 전환하고 write 가 진행되면 작성을 한다. 따라서 이와 같이 전혀 관련 없는 프로세스가 일회 작성하고 종료되는 방식으로 작동이 되면, RDWR권한으로 fifo 파일을 열었을 때 WR권한이 있는 프로세스가 존재하므로 대기하지만 RDONLY로 fifo 파일을 열면 대기하지 않고 계속 읽어드리므로 무한루프에 빠지게 된다.

(read(fd, msgbuf, MSGSIZ+1) <= 0) 조건으로 read하면 나타나는 현상: RDWR권한으로 열고 read하면 입력이 있을 때만 read하기 때문에 문제가 발생하지 않는다 하지만 RDONLY상태일 때 아무런 입력이 없으면 read의 반환값이 0 이 되므로 계속 에러처리 구문이 출력된다.