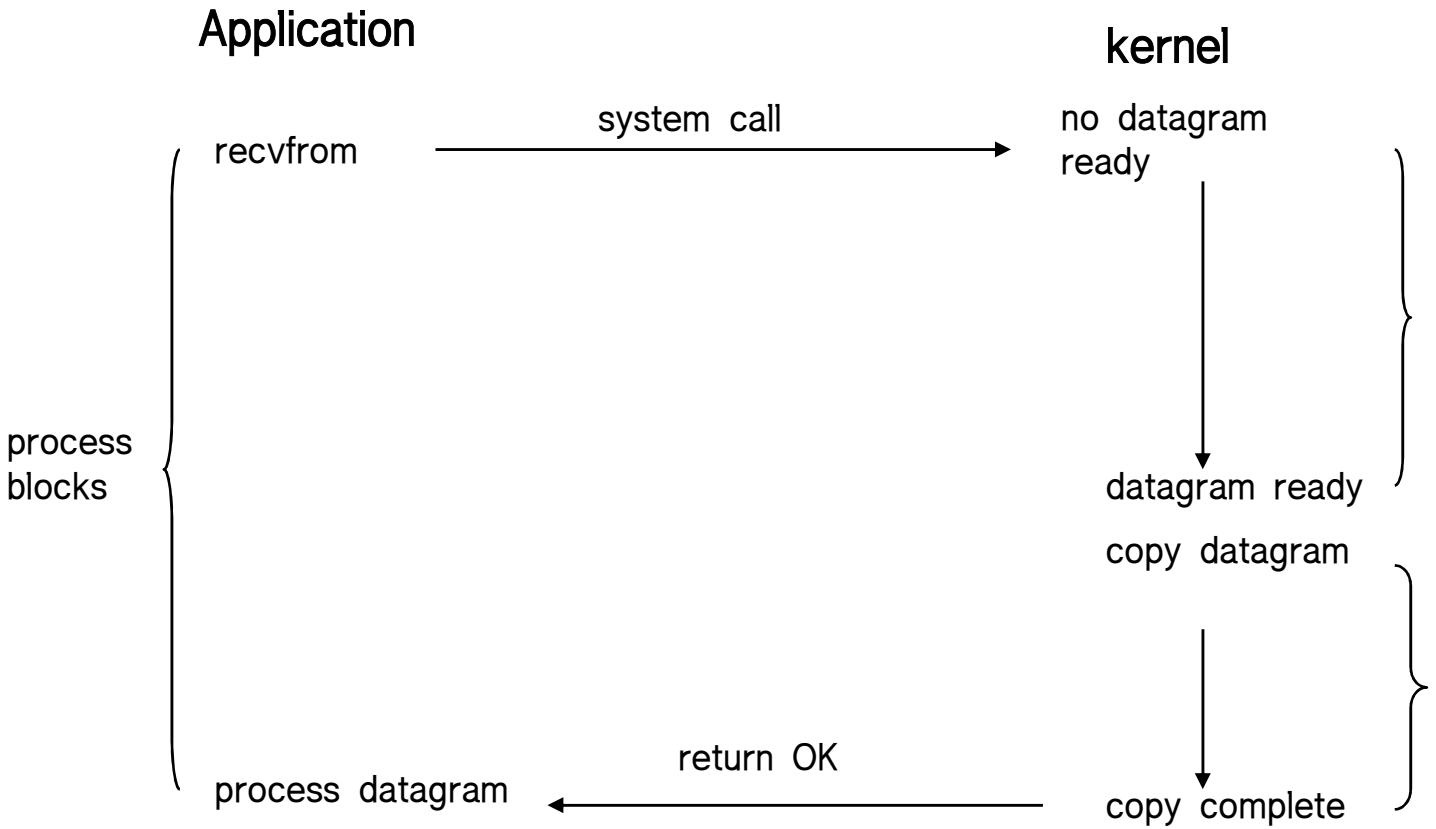
네트워크 프로그래밍

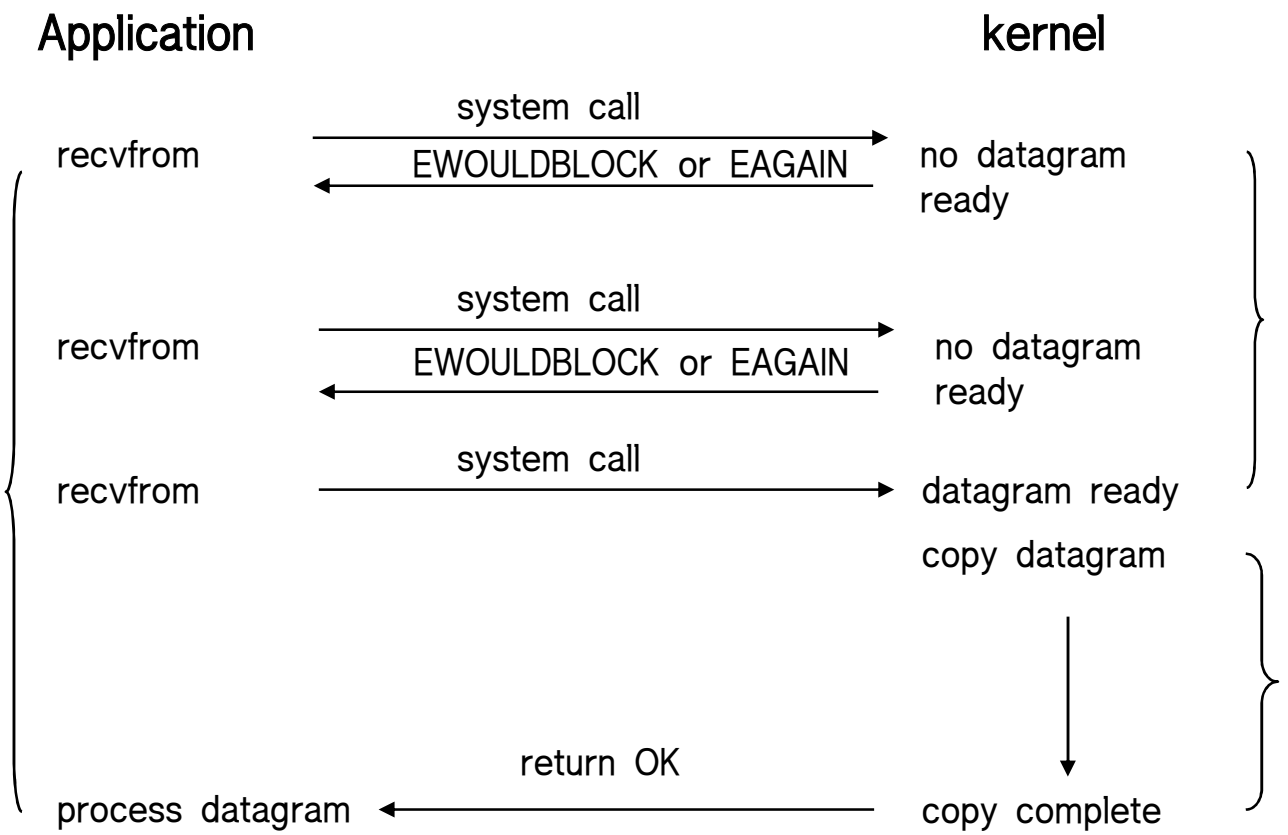
19. Nonblocking I/O

Blocking I/O


Application
kernel
system call
recvfrom
no datagram ready
process blocks
datagram ready
copy datagram
return OK
process datagram
copy complete

Nonblocking I/O

Nonblocking I/O



Nonblocking I/O

```
#include <fcntl.h>
#include <unistd.h>

int fcntl(int fd, int cmd, ...);
```

- fcntl 함수의 리턴 값
  - cmd에 따라 다름.
  - 그러나, 실패하면 -1을 리턴하고, errno를 설정

fcntl을 사용한 O_NONBLOCK 플래그의 추가/제거

```
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>

int setnonblock(int fd) {
    int fdflags;

    if ((fdflags = fcntl(fd, F_GETFL, 0)) == -1)
        return -1;
    fdflags |= O_NONBLOCK;
    if (fcntl(fd, F_SETFL, fdflags) == -1)
        return -1;
    return 0;
}
```

기존의 flag를 얻어오고
O_NONBLOCK플래그를 추가하고
set해줌
이때 추가하려면
반드시 |=를 이용

```
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>

int setblock(int fd) {
    int fdflags;

    if ((fdflags = fcntl(fd, F_GETFL, 0)) == -1)
        return -1;
    fdflags &= ~O_NONBLOCK;
    if (fcntl(fd, F_SETFL, fdflags) == -1)
        return -1;
    return 0;
}
```

뺄때는 &=~이용

개별적인 송수신 함수를 Nonblocking으로 만들 수 있는 방법

- **ssize_t send(int *sockfd*, const void **buf*, size_t *len*, int *flags*);**
- **ssize_t sendto(int *sockfd*, const void **buf*, size_t *len*, int *flags*, const struct sockaddr **dest_addr*, socklen_t *addrlen*);**
- **ssize_t recv(int *sockfd*, void **buf*, size_t *len*, int *flags*);**
- **ssize_t recvfrom(int *sockfd*, void **buf*, size_t *len*, int *flags*, struct sockaddr **src_addr*, socklen_t **addrlen*);**

이 두개는 udp통신할때사용했던것

- *flags*에 **MSG_DONTWAIT** 설정

Nonblocking I/O

- Nonblocking I/O 모델에서는
  - 작업이 완료되지 않으면,
    - -1을 리턴 & errno를 EWOULDBLOCK(EAGAIN)로 지정
    - 예외
      - TCP connect()의 경우
        - errno를 EINPROGRESS로 지정, 그러나 TCP three-way handshake는 계속된다.
        - 연결 성공 여부는 select()를 통해서 확인이 가능하다.

2번째인
syn, ack를 받아야
connect가 return되는데
》논블록의 경우
바로 리턴하기 때문에
select를 이용해서
》1번째에서 보낸syn에 대한
ack를 받을 경우 확인가능!

- Nonblocking 소켓 호출의 문제점
  - 함수의 호출이 언제 성공할지 모르기 때문에,
    - 성공할 때까지 주기적으로 호출을 시도(polling)해야 한다.

polling = = 반복문내에
read, write가 존재