Systems Programming

Spring 2023

11주차

- 11주차 ~ 14주차 진행 일정 안내

- 중간고사 2
  - claim 진행 방법 관련 안내
  - 배점 및 출제의도 안내

- Q&A
  - 14-concurrent-programming
  - 15-io-model

Q&A Session 진행 계획 (UPDATE)

- 11주차 (5/18)
  - 14-concurrent-programming
  - 15-io-model
- 12주차 (5/25)
  - 16-sync-basic
  - 17-sync-advanced
  - 17-sync-supplement
- **13주차 (6/1)**
  - **초청 세미나**
- 14주차 (6/8)
  - 18-optimization
  - 19-optimization-cache
  - 20-parallelism
- 기말고사 (6/15 목)

초청 세미나

- 출석체크 → 성적반영
  - 입실/퇴실 두 번 출석 체크 진행
  - 지정 좌석제 (etl 추후 공지)

- 일정
  - 6월 1일 목요일
  - 오후 2시 ~ 3시 15분
  - 302동 105호

중간고사 2

- 5/18 (목) 세션에 참석하지 않은 학생의 claim 받지 않음

- 채점 기준 추후 claim 시간에 조교들이 설명 예정
  - 기말고사 이전 채점 및 claim 진행 예정

- claim 진행 방식 변경
  - 채점 기준 확인
  - 본인 답안지 확인

  - 본인 답안의 특정 부분이 어떠한 이유로 점수가 수정되어야 하는지
  - 정해진 양식에 따라 작성하여
  - 조교들이 일괄 취합 (email, google form 등 양식 및 방식 추후 공지)

  - 조교들 내부 논의를 거쳐 claim 반영 여부 개인별 추후 확인

11주차

- 11주차 ~ 14주차 진행 일정 안내

- 중간고사 2
  - claim 진행 방법 관련 안내
  - 배점 및 출제의도 안내

- Q&A
  - 14-concurrent-programming
  - 15-io-model

14-concurrent-programming, 15-io-model

Questions

2) 14 Concurrent programming p11에서 Close(connfd)를 parent에서 수행해야만 하는 이유는 ref_cnt를 1감소시켜서 나중에 refcnt로 garbage collection을 하기 위함으로 이해했는데요, 그 이유 말고도 Accept를 했을 때 생긴 parent의 connfd와의 connection entry를 close(connfd)를 통해 Connection Table에서 Parent의 connfd와의 connection entry를 지움으로써 Client-Connfd의 connection entry만 남게 하기위한 의도가 있다고 이해했습니다. 내부 구현이 이런방식으로 진행되는게 맞는지 궁금합니다.

Process-Based Concurrent Server

```
int main(int argc, char **argv)
{
    int listenfd, connfd;
    int port = atoi(argv[1]);
    struct sockaddr_in clientaddr;
    int clientlen=sizeof(clientaddr);

    Signal(SIGCHLD, sigchld_handler);
    listenfd = Open_listenfd(port);
    while (1) {
        connfd = Accept(listenfd, (SA *) &clientaddr, &clientlen);
        if (Fork() == 0) {
            Close(listenfd); /* Child closes its listening socket */
            echo(connfd);    /* Child services client */
            Close(connfd);   /* Child closes connection with client */
            exit(0);         /* Child exits */
        }
        Close(connfd); /* Parent closes connected socket (important!) */
    }
}
```

Fork separate process for each client

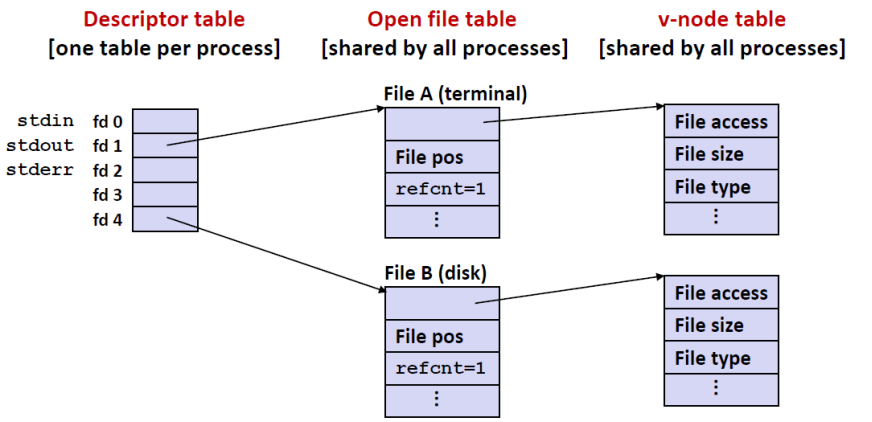
Does not allow any communication between different client handlers

Questions

2) 14 Concurrent programming p11에서 Close(connfd)를 parent에서 수행해야만 하는 이유는 ref_cnt를 1감소시켜서 나중에 refcnt로 garbage collection을 하기 위함으로 이해했는데요, 그 이유 말고도 Accept를 했을 때 생긴 parent의 connfd와의 connection entry를 close(connfd)를 통해 Connection Table에서 Parent의 connfd와의 connection entry를 지움으로써 Client-Connfd의 connection entry만 남게 하기위한 의도가 있다고 이해했습니다. 내부 구현이 이런방식으로 진행되는게 맞는지 궁금합니다.

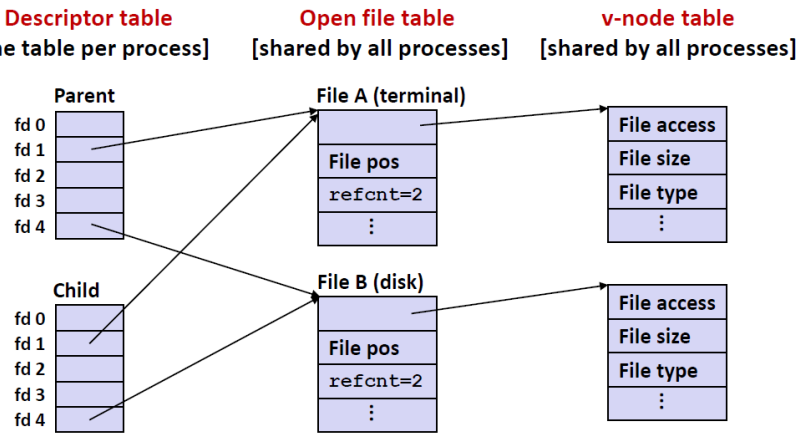
How Processes Share Files: Fork()

- A child process inherits its parent's open files
  - Note: situation unchanged by `exec` functions (use `fcntl` to change)
- *Before* fork() call:



How Processes Share Files: Fork()

- A child process inherits its parent's open files
- *After* fork():
  - Child's table same as parent's, and +1 to each refcnt



Connection	Host	Port	Host	Port
Listening	---	---	128.2.220.10	15213
cl1	128.2.192.34	50437	128.2.220.10	15213
cl2	128.2.205.225	41656	128.2.220.10	15213

Questions

1) 14 Concurrent programming p16에서 보면 Kernel이 Connection Table을 관리합니다.
만약 Host 128.2.192.32:50437에서 request가 왔으면 해당 테이블만으로는 적절한 Child process의 connfd를 찾아갈 수 없을 것 같은데, 실제 테이블에는 해당 connection에 맞는 Connfd의 주소값 entry가 더 있는 형태인건가요? 만약 아니라면 어떻게 적절한 connfd를 찾아가나요?

Connection	Host	Port	Host	Port
Listening	---	---	128.2.220.10	15213
cl1	128.2.192.34	50437	128.2.220.10	15213
cl2	128.2.205.225	41656	128.2.220.10	15213

Questions

```
1491 int __sys_socket(int family, int type, int protocol)
1492 {
1493     int retval;
1494     struct socket *sock;
1495     int flags;
1496
1497     /* Check the SOCK_* constants for consistency.  */
1498     BUILD_BUG_ON(SOCK_CLOEXEC != O_CLOEXEC);
1499     BUILD_BUG_ON((SOCK_MAX | SOCK_TYPE_MASK) != SOCK_TYPE_MASK);
1500     BUILD_BUG_ON(SOCK_CLOEXEC & SOCK_TYPE_MASK);
1501     BUILD_BUG_ON(SOCK_NONBLOCK & SOCK_TYPE_MASK);
1502
1503     flags = type & ~SOCK_TYPE_MASK;
1504     if (flags & ~(SOCK_CLOEXEC | SOCK_NONBLOCK))
1505         return -EINVAL;
1506     type &= SOCK_TYPE_MASK;
1507
1508     if (SOCK_NONBLOCK != O_NONBLOCK && (flags & SOCK_NONBLOCK))
1509         flags = (flags & ~SOCK_NONBLOCK) | O_NONBLOCK;
1510
1511     retval = sock_create(family, type, protocol, &sock);
1512     if (retval < 0)
1513         return retval;
1514
1515     return sock_map_fd(sock, flags & (O_CLOEXEC | O_NONBLOCK));
1516 }
1517
1518 SYSCALL_DEFINE3(socket, int, family, int, type, int, protocol)
"net/socket.c" 3774 lines --39%--
```

Questions

```
102 /**
103  *  struct socket - general BSD socket
104  *  @state: socket state (%SS_CONNECTED, etc)
105  *  @type: socket type (%SOCK_STREAM, etc)
106  *  @flags: socket flags (%SOCK_NOSPACE, etc)
107  *  @ops: protocol specific socket operations
108  *  @file: File back pointer for gc
109  *  @sk: internal networking protocol agnostic socket representation
110  *  @wq: wait queue for several uses
111  */
112 struct socket {
113     socket_state        state;
114
115     short           type;
116
117     unsigned long       flags;
118
119     struct file     *file;
120     struct sock     *sk;
121     const struct proto_ops  *ops;
122
123     struct socket_wq    wq;
124 };
include/linux/net.h
```

Questions

```
1332 /**
1333  *  __sock_create - creates a socket
1334  *  @net: net namespace
1335  *  @family: protocol family (AF_INET, ...)
1336  *  @type: communication type (SOCK_STREAM, ...)
1337  *  @protocol: protocol (0, ...)
1338  *  @res: new socket
1339  *  @kern: boolean for kernel space sockets
1340  *
1341  *  Creates a new socket and assigns it to @res, passing through LSM.
1342  *  Returns 0 or an error. On failure @res is set to %NULL. @kern must
1343  *  be set to true if the socket resides in kernel space.
1344  *  This function internally uses GFP_KERNEL.
1345  */
1346
1347 int __sock_create(struct net *net, int family, int type, int protocol,
1348                   struct socket **res, int kern)
1349 {
1350     int err;
1351     struct socket *sock;
1352     const struct net_proto_family *pf;
1353
1354     /*
1355      *      Check protocol is in range
1356      */
1357     if (family < 0 || family >= NPROTO)
1358         return -EAFNOSUPPORT;
1359     if (type < 0 || type >= SOCK_MAX)
1360         return -EINVAL;
1361
1362     /* Compatibility.
1363
1364        This uglymoron is moved from INET layer to here to avoid
1365        deadlock in module load.
1366      */
1367     if (family == PF_INET && type == SOCK_PACKET) {
1368         pr_info_once("%s uses obsolete (PF_INET,SOCK_PACKET)\n",
1369                      current->comm);
1370         family = PF_PACKET;
1371     }
1372
1373     err = security_socket_create(family, type, protocol, kern);
1374     if (err)
1375         return err;
1376
1377     /*
1378      *  Allocate the socket and allow the family to set things up. if
1379      *  the protocol is 0, the family is instructed to select an appropriate
1380      *  default.
1381      */
1382     sock = sock_alloc();
"net/socket.c" 3774 lines --36%--
```

Questions

```
411 static int sock_map_fd(struct socket *sock, int flags)
412 {
413     struct file *newfile;
414     int fd = get_unused_fd_flags(flags);
415     if (unlikely(fd < 0)) {
416         sock_release(sock);
417         return fd;
418     }
419
420     newfile = sock_alloc_file(sock, flags, NULL);
421     if (!IS_ERR(newfile)) {
422         fd_install(fd, newfile);
423         return fd;
424     }
425
426     put_unused_fd(fd);
427     return PTR_ERR(newfile);
428 }
429
430 /**
431  *  sock_from_file - Return the &socket bounded to @file.
432  *  @file: file
433  *  @err: pointer to an error code return
434  *
435  *  On failure returns %NULL and assigns -ENOTSOCK to @err.
436  */
437
438 struct socket *sock_from_file(struct file *file, int *err)
439 {
440     if (file->f_op == &socket_file_ops)
441         return file->private_data;  /* set in sock_map_fd */
442
443     *err = -ENOTSOCK;
444     return NULL;
445 }
446 EXPORT_SYMBOL(sock_from_file);
447
448 /**
449  *  sockfd_lookup - Go from a file number to its socket slot
"net/socket.c" 3774L, 95223C
```

Questions

```
567 /*
568  * Install a file pointer in the fd array.
569  *
570  * The VFS is full of places where we drop the files lock between
571  * setting the open_fds bitmap and installing the file in the file
572  * array.  At any such point, we are vulnerable to a dup2() race
573  * installing a file in the array before us.  We need to detect this and
574  * fput() the struct file we are about to overwrite in this case.
575  *
576  * It should never happen - if we allow dup2() do it, _really_ bad things
577  * will follow.
578  *
579  * NOTE: __fd_install() variant is really, really low-level; don't
580  * use it unless you are forced to by truly lousy API shoved down
581  * your throat.  'files' *MUST* be either current->files or obtained
582  * by get_files_struct(current) done by whoever had given it to you,
583  * or really bad things will happen.  Normally you want to use
584  * fd_install() instead.
585  */
586
587 void __fd_install(struct files_struct *files, unsigned int fd,
588         struct file *file)
589 {
590     struct fdtable *fdt;
591
592     rcu_read_lock_sched();
593
594     if (unlikely(files->resize_in_progress)) {
595         rcu_read_unlock_sched();
596         spin_lock(&files->file_lock);
597         fdt = files_fdtable(files);
598         BUG_ON(fdt->fd[fd] != NULL);
599         rcu_assign_pointer(fdt->fd[fd], file);
600         spin_unlock(&files->file_lock);
601         return;
602     }
603     /* coupled with smp_wmb() in expand_fdtable() */
604     smp_rmb();
605     fdt = rcu_dereference_sched(files->fdt);
606     BUG_ON(fdt->fd[fd] != NULL);
607     rcu_assign_pointer(fdt->fd[fd], file);
608     rcu_read_unlock_sched();
609 }
610
611 void fd_install(unsigned int fd, struct file *file)
612 {
613     __fd_install(current->files, fd, file);
614 }
615
616 EXPORT_SYMBOL(fd_install);
"fs/file.c" 1017 lines --55%--
```

Questions

17p에 port demultiplexing에 대한 설명이 나오는데 이것이 port multiplexing과 다른 점을 잘 인지하지 못했습니다. 찾아보니 역과정이라고 표현되기도 하는거같은데 둘 다 연결된 client에 따라 connection이 달라진다는 점을 이용하는 동일한 process인 것 같은데 조금 더 자세히 설명해주시면 감사하겠습니다.

View from Server's TCP Manager

Connection	Host	Port	Host	Port
Listening	---	---	128.2.220.10	15213
cl1	128.2.192.34	50437	128.2.220.10	15213
cl2	128.2.205.225	41656	128.2.220.10	15213

- **Port Demultiplexing**
  - TCP manager maintains separate stream for each connection
    - Each represented to application program as socket
    - New connections directed to listening socket
    - Data from clients directed to one of the connection sockets

Questions

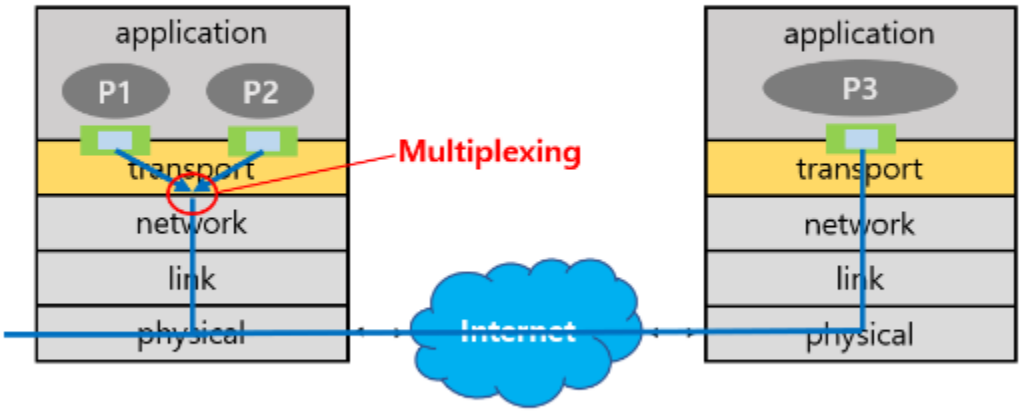
17p에 port demultiplexing에 대한 설명이 나오는데 이것이 port multiplexing과 다른 점을 잘 인지하지 못했습니다. 찾아보니 역과정이라고 표현되기도 하는거같은데 둘 다 연결된 client에 따라 connection이 달라진다는 점을 이용하는 동일한 process인 것 같은데 조금 더 자세히 설명해주시면 감사하겠습니다.

3. Multiplexing, Demultiplexing

어플리케이션들이 Transport Layer의 프로토콜을 이용하기 위해서는 프로세스마다 **Socket**을 이용해야 한다고 했다.하지만, 컴퓨터에는 여러개의 어플리케이션들이 돌아가고, 각각의 어플리케이션들은 하나 이상의 Socket을 생성할 수 있다. 따라서 Transport layer 입장에서는 여러개의 Socket에서 데이터들이 쏟아지는 형태가 된다.이렇게 여러 어플리케이션의 Socket들이 데이터를 송/수신하므로 필요한 개념이 바로 Multiplexing과 Demultiplexing이다.

1) Multiplexing

여러 어플리케이션의 Socket들로부터 들어오는 데이터를 수집하여, 패킷으로 만들어 하위 레이어로 전송하는 것



https://ddongwon.tistory.com/79

Multiplexing

Questions

17p에 port demultiplexing에 대한 설명이 나오는데 이것이 port multiplexing과 다른 점을 잘 인지하지 못했습니다. 찾아보니 역과정이라고 표현되기도 하는거같은데 둘 다 연결된 client에 따라 connection이 달라진다는 점을 이용하는 동일한 process인 것 같은데 조금 더 자세히 설명해주시면 감사하겠습니다.

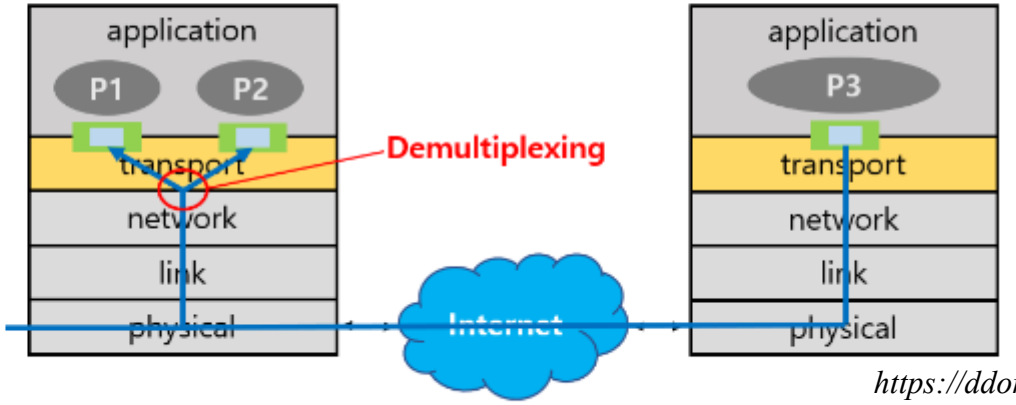
3. Multiplexing, Demultiplexing

어플리케이션들이 Transport Layer의 프로토콜을 이용하기 위해서는 프로세스마다 **Socket**을 이용해야 한다고 했다.하지만, 컴퓨터에는 여러개의 어플리케이션들이 돌아가고, 각각의 어플리케이션들은 하나 이상의 Socket을 생성할 수 있다. 따라서 Transport layer 입장에서는 여러개의 Socket에서 데이터들이 쏟아지는 형태가 된다.이렇게 여러 어플리케이션의 Socket들이 데이터를 송/수신하므로 필요한 개념이 바로 Multiplexing과 Demultiplexing이다.

2) Demultiplexing

하위 레이어로부터 수신된 패킷을 올바른 Socket으로 전송하여 올바른 어플리케이션에게 전송하는 것.

이때 정확한 어플리케이션의 Socket으로 전달해주기 위해 포트넘버를 활용한다.



https://ddongwon.tistory.com/79

Demultiplexing

Questions

22p에 보면 "• constant FD_SETSIZE defined by including <sys/select.h>, is the number of descriptors in the fd_set datatype.(1024)" 라는 내용이 나오는데 이 상수의 사용처를 모르겠습니다. 그냥 maxfdp1값이 FD_SETSIZE 보다 작아야 하므로 select함수를 호출할때 주의해라 이정도로 받아들이면 될까요?

Maxfdp1 argument

- specifies the number of descriptors to be tested.
- Its value is the maximum descriptor to be tested, plus one.(hence our name of maxfdp1)(example:fd1,2,5 => maxfdp1: 6)
- constant FD_SETSIZE defined by including <sys/select.h>, is the number of descriptors in the fd_set datatype.(1024)

Questions

```
SELECT(2)

NAME
       select, pselect, FD_CLR, FD_ISSET, FD_SET, FD_ZERO - synchronous I/O multiplexing

SYNOPSIS
       /* According to POSIX.1-2001, POSIX.1-2008 */
       #include <sys/select.h>

       /* According to earlier standards */
       #include <sys/time.h>
       #include <sys/types.h>
       #include <unistd.h>

       int select(int nfds, fd_set *readfds, fd_set *writefds,
                  fd_set *exceptfds, struct timeval *timeout);

       void FD_CLR(int fd, fd_set *set);
       int  FD_ISSET(int fd, fd_set *set);
       void FD_SET(int fd, fd_set *set);
       void FD_ZERO(fd_set *set);

       #include <sys/select.h>

       int pselect(int nfds, fd_set *readfds, fd_set *writefds,
                   fd_set *exceptfds, const struct timespec *timeout,
                   const sigset_t *sigmask);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       pselect(): _POSIX_C_SOURCE >= 200112L
```

"include/uapi/linux/posix_types.h"

```
1  /* SPDX-License-Identifier: GPL-2.0 WITH Linux-syscall-note */
2  #ifndef _LINUX_POSIX_TYPES_H
3  #define _LINUX_POSIX_TYPES_H
4
5  #include <linux/stddef.h>
6
7  /*
8   * This allows for 1024 file descriptors: if NR_OPEN is ever grown
9   * beyond that you'll have to change this too. But 1024 fd's seem to be
10  * enough even for such "real" unices like OSF/1, so hopefully this is
11  * one limit that doesn't have to be changed [again].
12  *
13  * Note that POSIX wants the FD_CLEAR(fd,fdsetp) defines to be in
14  * <sys/time.h> (and thus <linux/time.h>) - but this is a more logical
15  * place for them. Solved by having dummy defines in <sys/time.h>.
16  */
17
18 /*
19  * This macro may have been defined in <gnu/types.h>. But we always
20  * use the one here.
21  */
22 #undef __FD_SETSIZE
23 #define __FD_SETSIZE    1024
24
25 typedef struct {
26     unsigned long fds_bits[__FD_SETSIZE / (8 * sizeof(long))];
27 } __kernel_fd_set;
28
29 /* Type of a signal handler.  */
30 typedef void (*__kernel_sighandler_t)(int);
31
32 /* Type of a SYSV IPC key.  */
33 typedef int __kernel_key_t;
34 typedef int __kernel_mqd_t;
35
36 #include <asm/posix_types.h>
37
38 #endif /* _LINUX_POSIX_TYPES_H */
```

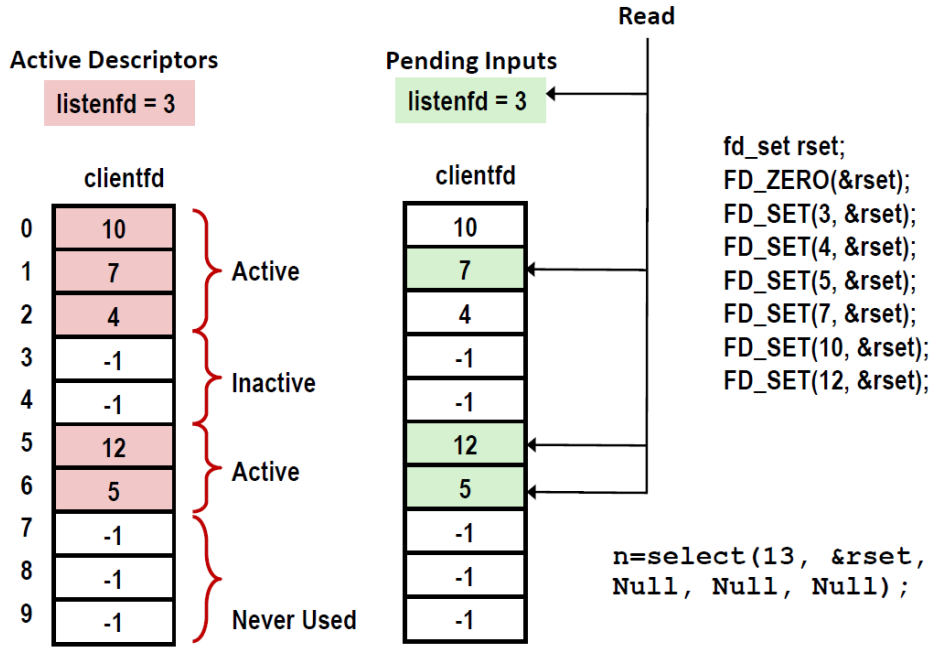
Questions

```
Text string: FD_SETSIZE

  File          Line
0 posix_types.h   22 #undef __FD_SETSIZE
1 posix_types.h   23 #define __FD_SETSIZE 1024
2 posix_types.h   26 unsigned long fds_bits[__FD_SETSIZE / (8 * sizeof(long))];
3 nolibc.h       183 #define FD_SETSIZE 256
4 nolibc.h       184 typedef struct { uint32_t fd32[FD_SETSIZE/32]; } fd_set;
5 nolibc.h      2447 if (fd < 0 || fd >= FD_SETSIZE)
6 nettest.c     1262 rc = select(FD_SETSIZE, NULL, &wfd, NULL, tv);


Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:

[0] 0:bash- 1:bash*                                  "hexa2" 20:34 17-May-23
```

Questions

39p에 그림을 강의를 들으며 listenfd는 3, 그리고 각 배열에 있는 번호들 (4~12)는 clientfd라고 이해했습니다. 하지만 그림에 나와있는 배열(0, 1, 2 ... 9번까지 indexing된) 이 무엇을 나타내는지 모르겠으며 -1로 비어있는 공간도 정확히 왜 존재하는지 이해하지 못했고 왜 connfd가 아닌 clientfd인지도 잘 모르겠습니다.

I/O Multiplexed Event Processing



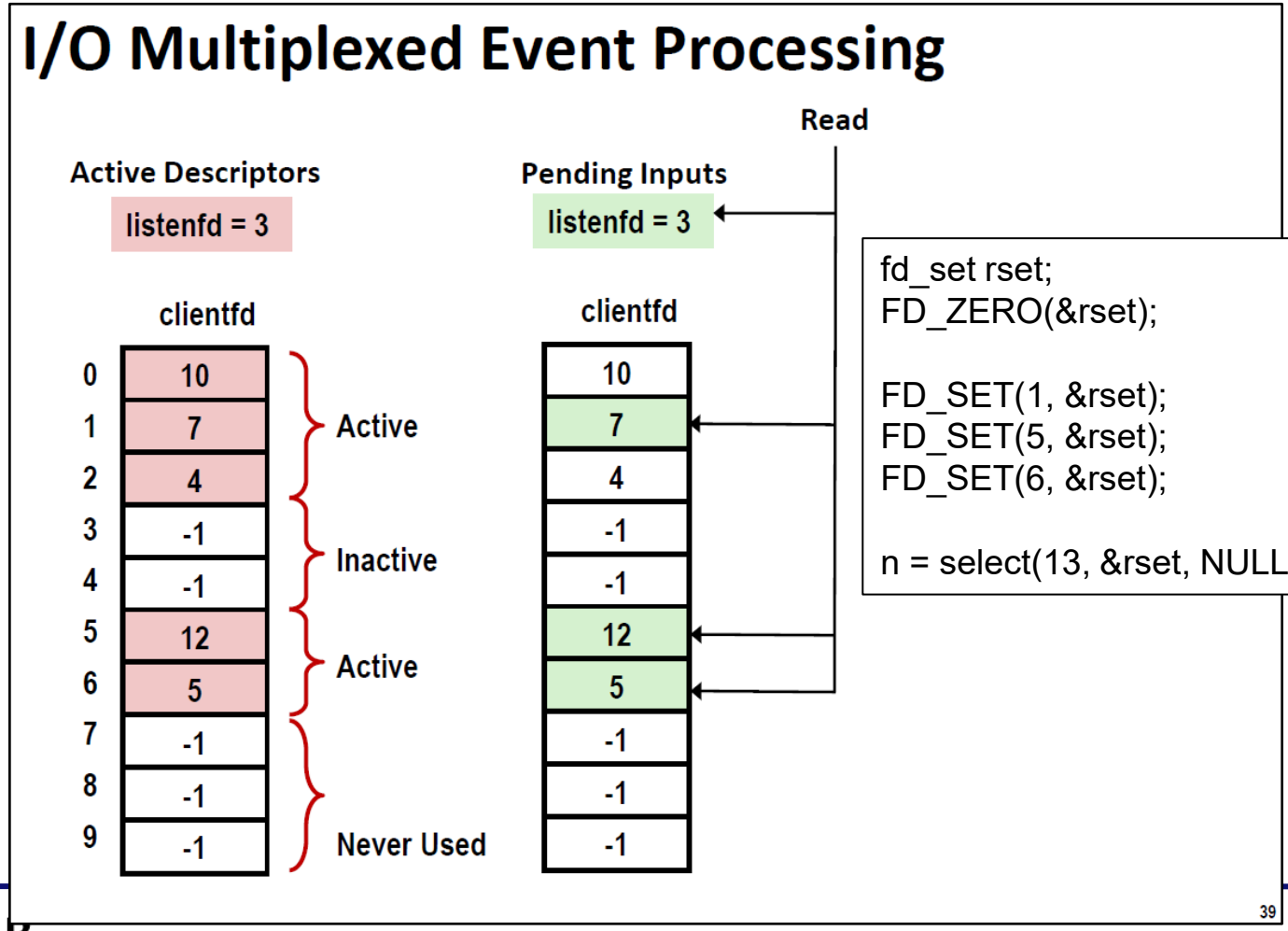
39

Event-Based Concurrent Servers Using I/O Multiplexing

- **Use library functions to construct scheduler within single process**
- **Server maintains set of active connections**
  - Array of connfd's + listenfd
- **Repeat:**
  - Determine which connections have pending inputs
  - If listenfd has input, then accept connection
    - Add new connfd to array
  - Service all connfd's with pending inputs
- **Details in book (next lecture)**

Questions

39p에 그림을 강의를 들으며 listenfd는 3, 그리고 각 배열에 있는 번호들 (4~12)는 clientfd라고 이해했습니다. 하지만 그림에 나와있는 배열(0, 1, 2 ... 9번까지 indexing된) 이 무엇을 나타내는지 모르겠으며 -1로 비어있는 공간도 정확히 왜 존재하는지 이해하지 못했고 왜 connfd가 아닌 clientfd인지도 잘 모르겠습니다.



```
SELECT(2)

NAME
       select, pselect, FD_CLR, FD_ISSET, FD_SET, FD_ZERO - synchronous I/O multiplexing
```

```
EXAMPLE
       #include <stdio.h>
       #include <stdlib.h>
       #include <sys/time.h>
       #include <sys/types.h>
       #include <unistd.h>

       int
       main(void)
       {
           fd_set rfds;
           struct timeval tv;
           int retval;

           /* Watch stdin (fd 0) to see when it has input. */

           FD_ZERO(&rfds);
           FD_SET(0, &rfds);

           /* Wait up to five seconds. */

           tv.tv_sec = 5;
           tv.tv_usec = 0;

           retval = select(1, &rfds, NULL, NULL, &tv);
           /* Don't rely on the value of tv now! */

           if (retval == -1)
               perror("select()");
           else if (retval)
               printf("Data is available now.\n");
               /* FD_ISSET(0, &rfds) will be true. */
           else
               printf("No data within five seconds.\n");

           exit(EXIT_SUCCESS);
       }
```

```
char buf[30];
fd_set readfds;
FD_ZERO(&readfds);
int fd[3];

while (true)
{
      for(int i=0;i<3;i++) {
            fd[i] = open(pipes[i], O_RDONLY | O_NONBLOCK);
            FD_SET(fd[i], &readfds);
      }
      int state = select(fd[2]+1, &readfds, NULL, NULL, NULL);
      if(state==-1) break;
      else if(state == 0) continue;

      for(int i=0;i<3;i++){
            if(fd[i]==-1) continue;
            if(FD_ISSET(fd[i], &readfds)){
                  read(fd[i], buf, sizeof(buf));
                  printf("%d pipe input: %s", i+1, buf);
            }
      }

      for(int i=0;i<3;i++) close(fd[i]);
}
```

https://velog.io/@leaps/System-Programming-select-poll-epoll

Questions

17p select4를 보면 "The wait during select can be interrupted by signals (first two ways)"라고 되어있는데 여기서 지칭하는 first two ways가 뭔지 정확히 이해하지 못했습니다.

select 4

- The wait during *select* can be interrupted by signals (first two ways)
- Exception conditions
  - The arrival of out-of-band data

Questions

24p를 보면 socket이 write할 준비가 되었다는 조건에 "A socket using a non-blocking connect has completed the connection, or the connect has failed"라고 되어있는데 이 조건이 writing이 준비된 것과 어떤 관련이 있는지 강의에서도 잘 설명되지 않는 것 같아 질문드립니다.

Conditions for Readiness 2

- A socket is ready for writing if any of the following conditions is true:
  - Available space in the socket send buffer is greater than the low-water mark(default 2048) and the socket is connected or does not require a connection (UDP)
  - The write-half of the connection is closed (SIGPIPE)
  - A socket using a non-blocking connect has completed the connection, or the connect has failed
  - A socket error is pending
- A socket has an exception condition pending if there exists out-of-band data for the socket.

Questions

24p를 보면 socket이 write할 준비가 되었다는 조건에 "A socket using a non-blocking connect has completed the connection, or the connect has failed"라고 되어있는데 이 조건이 writing이 준비된 것과 어떤 관련이 있는지 강의에서도 잘 설명되지 않는 것 같아 질문드립니다.

Nonblocking I/O

- When an I/O cannot be completed, the process is not put to sleep, but returns with an error (EWOULDBLOCK)
- Waste of CPU time

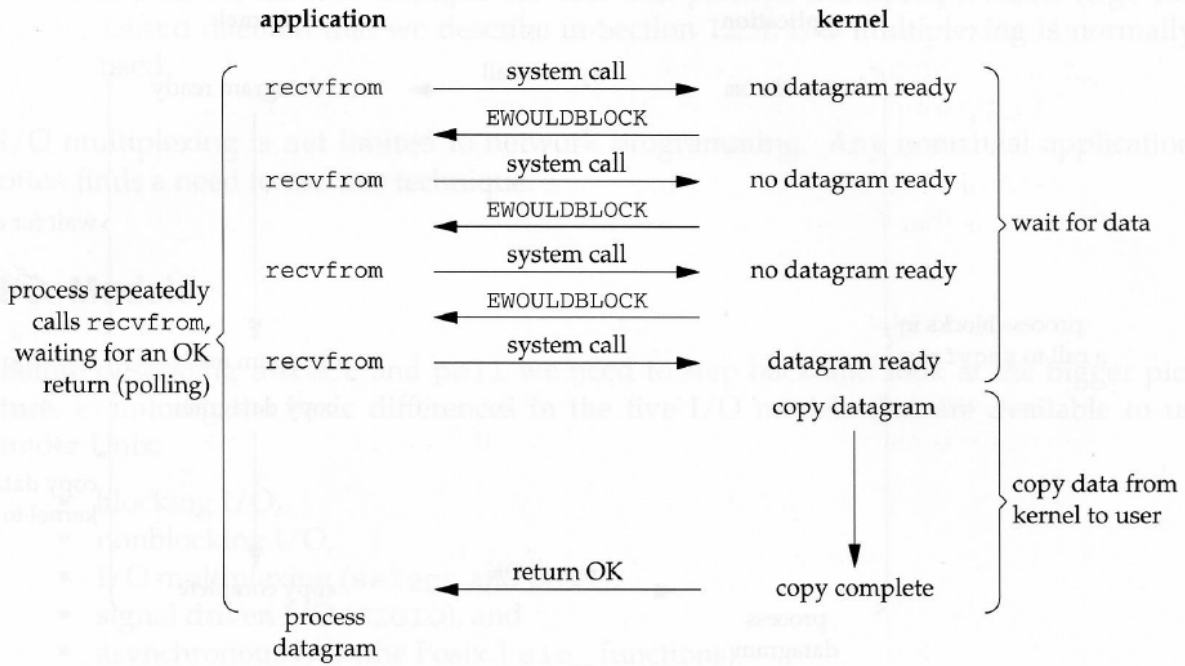


Figure 6.2 Nonblocking I/O model.