Systemy Operacyjne

KOMUNIKACJA MIDZYPROCESOWA

Zajęcia 6 – KOMUNIKACJA MIĘDZYPROCESOWA

- 1 Tworzenie prostych łączy jednokierunkowych
- 2 Praca z łączami komunikacyjnymi
- 3 Ćwiczenia dot. łączy nienazwanych
  - 3.1 Komunikacja z kilkoma procesami
  - 3.2 Przekazywanie danych przez kilka łączy
  - 3.3 Przykładowy potok z Unixa

Przypomnienie

- Rodzaje plików
  - Pliki
  - Kolejki FIFO / pipe

- 3 standardowe deskryptory:
  - man 3 stdin
  - 0 -STDIN_FILENO
  - 1- STDOUT_FILENO
  - 2- STDERR_FILENO

- pid_t pid = fork()
  - 0 – proces macierzysty
  - >0 – proces potomny

- pipe w bash
  - | - przekazywanie wy do wejścia (procesy)
  - np. ls | sort –nr

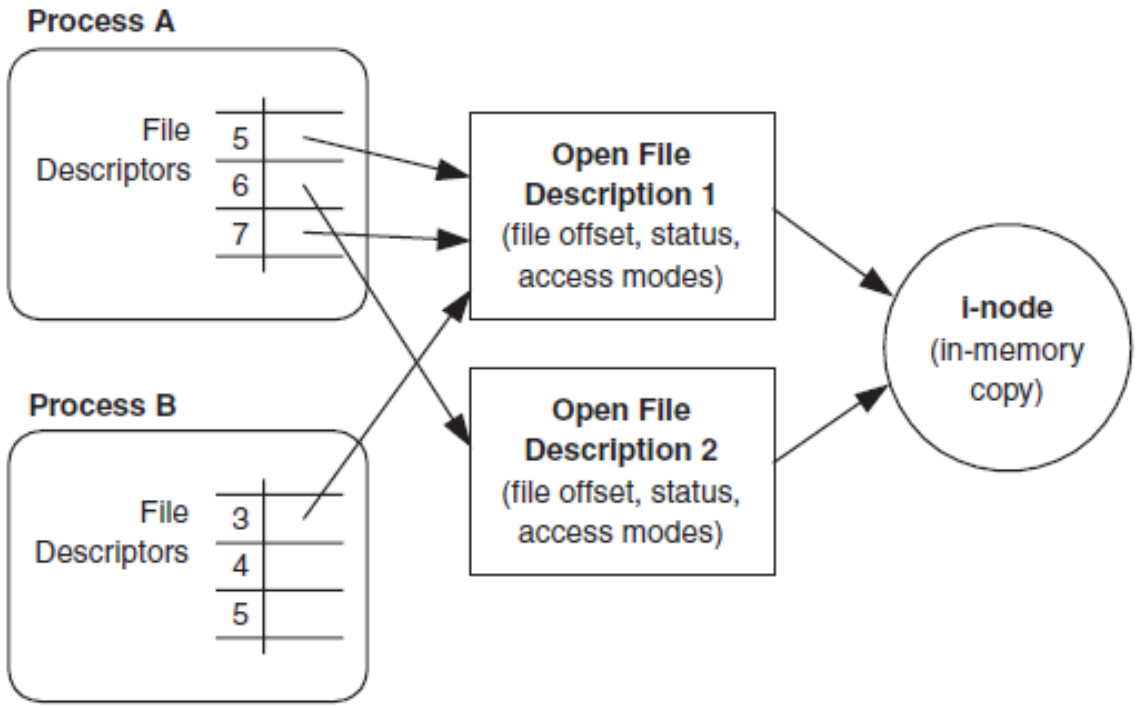


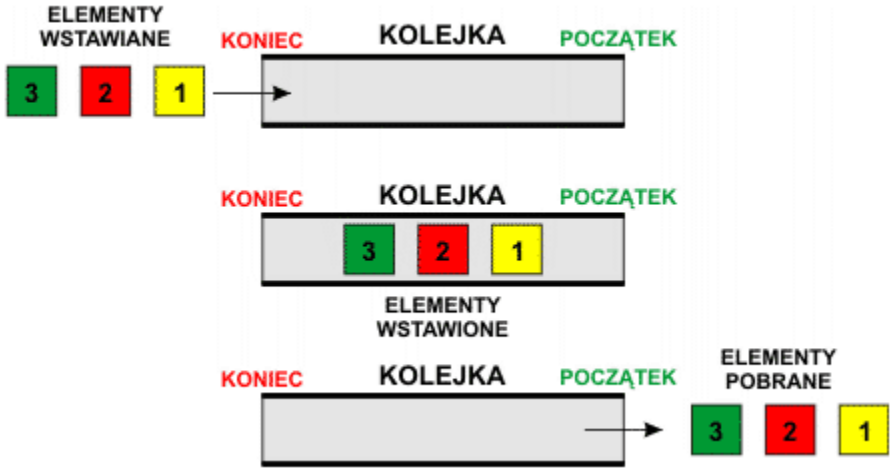
Figure 2.1 File descriptors, open file descriptions, and i-nodes.

Łącza nienazwane (pipe)

- man 7 pipe

- pipe
  - int pipe(int pfd[2]);
  - 1 – zapis, 0 – odczyt

- Operacje we/wy na deskryptorach
  - write
  - read
  - close
  - fstat
  - fpathconf(fd[0], _PC_PIPE_BUF)
  - dup
  - lseek



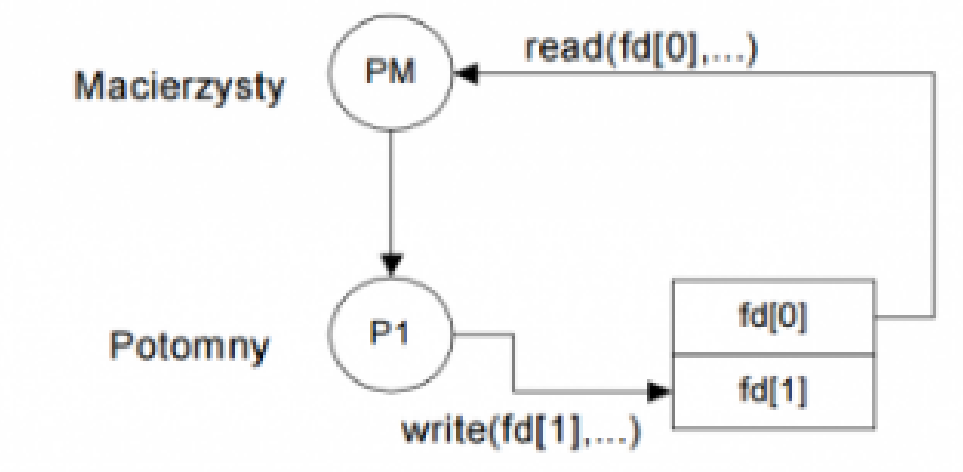
kolejka FIFO

http://eduinf.waw.pl/inf/alg/001_search/0105.php

Łącza nienazwane

int pfd[2]

int pipe(pfd);



- Błędy zwracane przez pipe
  man 2 pipe

- synchroniczny odczyt i zapis
- O_NONBLOCK - brak zapewnionej ciągłości w zapisie danych (np. w przypadku przepełnienia), write i read nie są blokowane
- PIPE_BUF – wielkość bufora – jeśli zostanie przepełniony to zwracany jest błąd przy zapisie

Synchroniczność

```
20    char * str  = ":test 1:";
21    char * str2 = "& msg2 &";
```

```
52   if (pid == 0) {
53
54        write(pfd[1], str2, strlen(str2));
55    } else if (pid >0 ) {
56
57        write(pfd[1], str, strlen(str));
58 //       sleep(1);
59
60        nread=read(pfd[0],buf, sizeof(buf));
61        buf[nread] = 0;
```

```
52   if (pid == 0) {
53
54        write(pfd[1], str2, strlen(str2));
55    } else if (pid >0 ) {
56
57        write(pfd[1], str, strlen(str));
58        sleep(1);
59
60        nread=read(pfd[0],buf, sizeof(buf));
61        buf[nread] = 0;
```

```
abasiura@galaxy:~/SO_zajecia/6/6_1$ gcc -o p test.c && p
:test1: (7 bytes)
PIPE 4096
PIPE 4096
```

```
abasiura@galaxy:~/SO_zajecia/6/6_1$ gcc -o p test.c && p
PIPE 4096
:test1:& msg 2& (15 bytes)
PIPE 4096
```

O_NONBLOCK - flaga

```
13    size_t nread;
...
16    pipe(pfd);
17    int flags = fcntl(pfd[1], F_GETFL);
18    flags |= O_NONBLOCK;
19    if (fcntl(pfd[1], F_SETFL, flags)) {
20          perror("cos nie tak");
21    }
22
23
24    flags = fcntl(pfd[0], F_GETFL);
25    flags |= O_NONBLOCK;
26    if (fcntl(pfd[0], F_SETFL, flags)) {
...

66   if (pid == 0) {
67
68       write(pfd[1], str2, strlen(str2));
69    } else if (pid >0 ) {
70
71       write(pfd[1], str, strlen(str));
72       //sleep(1);
73
74       nread=read(pfd[0],buf, sizeof(buf));
75       if (nread>0) buf[nread-1]=0;
76       (nread>0)?printf("%s (%ld bytes)\n",buf,(long)nread):printf("No data %ld \n", (long)nread);
77
78
79       nread=read(pfd[0],buf, sizeof(buf));
80       if (nread>0) buf[nread-1]=0;
81       if (nread!=0 && nread!=-1) {
82            printf("%s (%ld bytes)\n",buf,(long)nread);
83       } else {
84           printf("No data %ld \n",(long) nread);
85       }
86
```

```
abasiura@galaxy:~/SO_zajecia/6/6_1$ gcc -o p test.c && p
:test1 (7 bytes)
No data -1
PIPE 4096
PIPE 4096
```

Łącza nienazwane

- fstat(pfd[0]);

```
16    pipe(pfd);
17
18    pid_t pid = fork();
19
20    char * str ="UR Beautiful pipe example:1";
21    char * str2 ="UR Beautiful 2";
22
23    if (pid == 0) {
24
25
26        struct stat sb;
27        fstat(pfd[1], &sb);
28
29        printf("File type:                ");
30        switch (sb.st_mode & S_IFMT) {
31            case S_IFBLK:  printf("block device\n");            break;
32            case S_IFCHR:  printf("character device\n");        break;
33            case S_IFDIR:  printf("directory\n");               break;
34            case S_IFIFO:  printf("FIFO/pipe\n");               break;
35            case S_IFLNK:  printf("symlink\n");                 break;
36            case S_IFREG:  printf("regular file\n");            break;
37            case S_IFSOCK: printf("socket\n");                  break;
38            default:       printf("unknown?\n");                break;
39            }
40        printf("I-node number:            %ld\n", (long) sb.st_ino);
41        printf("Mode:                     %lo (octal)\n",
42                    (unsigned long) sb.st_mode);
```

```
abasiura@galaxy:~/SO_zajecia/6/6_1$ gcc -o p test.c
abasiura@galaxy:~/SO_zajecia/6/6_1$ p
UR Beautiful pipe example:1 (27 bytes)
PIPE 4096
File type:                FIFO/pipe
I-node number:            898198810
Mode:                     10600 (octal)
Link count:               1
Ownership:                UID=7668   GID=434
```

- W zależności od wersji UNIX pipe może być też dwu kierunkowy

Dup, Dup2

- Kopiowanie deskryptora
  - dup(fd) : fcntl(fd, F_DUPFD, 0);
  - dup2(fd,0): close(0); fcntl(fd, F_DUPFD, fd2);

- Możliwość „podmienienie” standardowego deskryptora

```
37          dup2(pfdout1[0], STDIN_FILENO);
38          dup2(pfdin1[1], STDOUT_FILENO);
39          execlp("grep","grep","pipe","-c",NULL);
```

- Zamykamy „niepotrzebne” deskryptory: pfdout1, pfdin1

Ćwiczenia

- **1.3-4 – tworzenie łączy**
- **2.2 – praca z łączami (dup, dup2)**

- 3.1 – komunikacja z kilkoma procesami (+)
- 3.2 – przekazywanie danych przez łącze (+)
- 3.3 – przykładowy potok (+)

- Dla zainteresowanych (+/ +)

M.J.Rochkind - Programowanie w systemie UNIX dla zaawansowanych (*Advanced UNIX Programming*). Wydanie 3PL / 2EN

Proszę przeczytać rozdział:

- Rozdział 7. Zaawansowana komunikacja międzyprocesowa (podrozdziały 7.1 – 7.7)