Politechnika Świętokrzyska w Kielcach **Wydział Elektrotechniki, Automatyki i Informatyki**	
Programowanie Współbieżne laboratorium	
Temat: **Łącza FIFO** (laboratorium nr 4)	Autor: **Mateusz Snoch** Grupa: **3ID11A**
Data odbycia się laboratorium : **09.11.2017**	Data sporządzenia sprawozdania: **10.11.2017**

Przydatne dane:

open – otwieranie połączenia między plikiem a deskryptorem pliku (w tym przypadku połączenie FIFO)
mknod – tworzenie nowego łącza
write – zapisywanie danych do łącza z buforu
read – czytanie danych z łącza do buforu
unlink – usunięcie utworzonej kolejki

Zadanie 2

Utworzyć przez *mknod /tmp/fifo.1 p* kolejkę fifo.
Sprawdzić przez *ls -la* czy plik powstał i jakiego jest typu
Przepisać program który zapisuje do kolejki prosty komunikat
Napisać analogiczny program który czyta z kolejki komunikatów
Uruchomić program czytający następnie piszący a potem na odwrót

Piszący:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>

#include <errno.h>
extern int errno;
#define FIFO1 "/tmp/fifo.1"
main()
{
        char buff[]="ala ma kota a kot ma ale";
        int fifo;
        if ((fifo = open(FIFO1, O_WRONLY )) < 0)
                    perror("nie moze otworzyc fifo1 do pisania");
        if (strlen(buff) != write(fifo,buff,strlen(buff)))
                    perror("blad zapisu do fifo");
        exit(0);
}
```

Czytający:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>

#include <errno.h>
extern int errno;
#define FIFO1 "/tmp/fifo.1"
main()
{
        char buff[100];
        int fifo;
        if ((fifo = open(FIFO1, O_RDONLY )) < 0)
                    perror("nie moze otworzyc fifo1 do odbierania");
        int n=read(fifo,buff,sizeof(buff));
                if(n<=0)
```

```
                    perror("blad zapisu do fifo");
                else{
                        write(1,buff,n);
                        printf("\n");
                }
        return 0;
}
```

```
student@st06-lab319:~/Pulpit$ mknod /tmp/fifo.1 p
student@st06-lab319:~/Pulpit$ cd ..
student@st06-lab319:~$ cd ..
student@st06-lab319:/HOME$ cd tmp
bash: cd: tmp: Nie ma takiego pliku ani katalogu
student@st06-lab319:/HOME$ cd tmp/
bash: cd: tmp/: Nie ma takiego pliku ani katalogu
student@st06-lab319:/HOME$ cd ..
student@st06-lab319:/$ cd tmp/
student@st06-lab319:/tmp$ ls -la
razem 28
drwxrwxrwt  6 root    root    4096 lis  9 19:21 .
drwxr-xr-x 25 root    root    4096 wrz 29  2014 ..
prw-------  1 student student    0 lis  9 19:21 fifo.1
drwxrwxrwt  2 root    root    4096 lis  9 19:04 .ICE-unix
drwx------  2 student student 4096 sty  1  1970 orbit-student
drwx------  2 student student 4096 lis  9 19:04 ssh-T1p5TcZeTfY2
-rw-------  1 student student    0 lis  9 19:04 unity_support_test.0
-r--r--r--  1 root    root      11 lis  9 19:02 .X0-lock
drwxrwxrwt  2 root    root    4096 lis  9 19:02 .X11-unix
```

Zadanie 3

Przy funkcji open zastosować flagę O_NDELAY, najpierw przy jednym programie potem przy drugim
Uruchomić pierwszy potem drugi
Uruchomić drugi potem pierwszy
Porównać wyniki. Co daje O_NDELAY?

1. Otwarcie kolejki FIFO tylko do czytania, a żaden proces nie otworzył tej kolejki do pisania:
Bez O_NDELAY: Czekanie, aż proces otworzy kolejkę FIFO do pisania
Z O_NDELAY: Powrót natychmiast bez sygnalizowania błędu

2. Otwarcie kolejkę FIFO tylko do pisania, a żaden proces nie otworzył tej kolejki do czytania:
Bez O_NDELAY: Czekanie, aż proces otworzy kolejkę FIFO do czytania
Z O_NDELAY: Powrót natychmiast, sygnalizowanie błędu, w errno umieszczenie stałą ENXIO

Zadanie 4

Do programu piszącego dodać funkcje tworzące kolejkę fifo a do odbierającego usuwającą ją.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>

#include <errno.h>
extern int errno;
#define FIFO1 "/tmp/fifo.1"
main()
{
        if(mknod(FIFO1,0666|S_IFIFO,0)<0){
                perror("nie mozna stworzyc fifo");
        }
        char buff[]="ala ma kota a kot ma ale";
        int fifo;
        if ((fifo = open(FIFO1, O_WRONLY /*| O_NDELAY*/ )) < 0)
                    perror("nie moze otworzyc fifo1 do pisania");
        if (strlen(buff) != write(fifo,buff,strlen(buff)))
                    perror("blad zapisu do fifo");
        exit(0);
}
```

Czytający:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>

#include <errno.h>
extern int errno;
#define FIFO1 "/tmp/fifo.1"
main()
{
        char buff[100];
        int fifo;
        if ((fifo = open(FIFO1, O_RDONLY/*|O_NDELAY*/ )) < 0)
                    perror("nie moze otworzyc fifo1 do odbierania");
        int n=read(fifo,buff,sizeof(buff));
                if(n<=0)
                    perror("blad zapisu do fifo");
                else{
                        write(1,buff,n);
                        printf("\n");
                }
                unlink(FIFO1);
        return 0;
}
```

Wnioski

Na czwartych zajęciach laboratoryjnych dowiedziałem się jak tworzyć kolejkę FIFO. Zaobserwowałem różnice między FIFO a łączem PIPE. Nauczyłem się tworzyć przykładowe połączenie i poznałem zasady, jakimi się kieruje. Pierwsze cztery zadania nie sprawiły mi problemu, jednak na komunikator zabrakło mi czasu. Zadania nie przysporzyły mi większych trudności.