

# Wstęp / Operacje na plikach

## Systemy Operacyjne 2

Piotr Zierhoffer

5 listopada 2011

Piotr Zierhoffer

**[piotr.zierhoffer@cs.put.poznan.pl](mailto:piotr.zierhoffer@cs.put.poznan.pl)**

pokój 2.7.2 (BTiCW)

[www.cs.put.poznan.pl/pzierhoffer](http://www.cs.put.poznan.pl/pzierhoffer)

telefon: (+48) 61 665 29 41

konsultacje: poniedziałek, 9:45 – 11:15

- obecność na zajęciach obowiązkowa
- kolokwium
- zadania domowe
- aktywność w trakcie zajęć

- 1 Wstęp / obsługa plików / procesy
- 2 Potoki / kolejki FIFO
- 3 Mechanizmy IPC:
  - 1 Kolejki komunikatów
  - 2 Pamięć współdzielona
  - 3 Semafor
- 4 Synchronizacja, **kolokwium**

## Materiały:

- <http://www.cs.put.poznan.pl/akobusinska/sop2.html>
- <http://www.cs.put.poznan.pl/csobaniec/edu/sop/sop2.html>
- <http://www.cs.put.poznan.pl/dwawrzyniak/>
- <http://www.cs.put.poznan.pl/sop/>

## Serwer:

- [\[unixlab|polluks\].cs.put.poznan.pl](http://[unixlab|polluks].cs.put.poznan.pl)

## Opis funkcji:

- [www.refcards.com](http://www.refcards.com)
- [cplusplus.com/reference/](http://cplusplus.com/reference/)

- Kompilacja:
  - `gcc program.c`
  - `gcc program.c -o program`
  - `gcc -Wall program.c -o program`
  - `gcc -g program.c -o program`
- Uruchomienie: `./program`
- Pomoc systemowa:
  - sekcja 2 — wywołania systemowe (`man 2 write`)
  - sekcja 3 — funkcje biblioteczne (`man 3 printf`)

```
#include <stdio.h>
int main(int argc, char **argv)
{
    int i;
    for(i = 0; i < argc; ++i)
    {
        printf("Argument[%d]: %s\n", i, argv[i]);
    }
    return 0;
}
```

- Identyfikacja pliku:
  - stream (FILE \*)
    - funkcje z biblioteki glibc, zgodne z ANSI C, wsparcie buforowania i formatowania
  - deskryptor pliku (int)
    - odwołania systemowe (syscall), bez bajerów
- W systemach UNIX:
  - tablica otwartych deskryptorów
  - standardowo:
    - stdin — 0
    - stdout — 1
    - stderr — 2



- Funkcja:

```
int open(const char *pathname, int flags)
```

```
int open(const char *pathname, int flags, mode_t mode)
```

- Parametry:

- `pathname` — nazwa pliku,

- `flags` — tryb otwarcia:

- `O_WRONLY`

- `O_RDONLY`

- `O_RDWR`

- `O_APPEND`

- `O_CREAT`

- `O_TRUNC`

- ...

- `mode` — prawa dostępu, np. 0642 (lub odpowiednie flagi)

- Wartość zwracana:

- deskryptor pliku lub -1

- Funkcja

```
int creat(const char *pathname, mode_t mode)
```

- Parametry:

- `pathname` — nazwa pliku
- `mode` — prawa dostępu, np. 0642 (lub odpowiednie flagi)

- Wartość zwracana:

- deskryptor pliku lub -1

- Ekwiwalent:

```
open(pathname, O_WRONLY|O_CREAT|O_TRUNC, mode)
```

## Zamykanie:

- Funkcja  
`int close(int fd)`
- Parametry:
  - `id` — deskryptor pliku
- Wartość zwracana:
  - 0 lub -1

## Usuwanie:

- Funkcja  
`int unlink(const char *pathname)`
- Parametry:
  - `pathname` — nazwa pliku
- Wartość zwracana:
  - 0 lub -1

- Funkcja:  
`ssize_t read(int fd, void *buf, size_t count)`
- Parametry:
  - `fd` — deskryptor pliku, z którego następuje odczyt danych,
  - `buf` — adres bufora, w którym zostaną umieszczone odczytane dane,
  - `count` — maksymalna liczba bajtów do odczytu z pliku (należy uważać, by nie była większa od rozmiaru bufora).
- Wartość zwracana:
  - ilość odczytanych bajtów lub -1
  - 0 oznacza koniec pliku/brak dalszych danych

- Funkcja:  
`ssize_t write(int fd, const void *buf, size_t count)`
- Parametry:
  - `fd` — deskryptor pliku, do którego następuje zapis danych,
  - `buf` — adres bufora zawierającego dane do zapisania
  - `count` — maksymalna liczba bajtów do zapisania
- Wartość zwracana
  - ilość zapisanych bajtów lub -1

Powielenie deskryptora:

- `int dup(int oldfd)`
- `int dup2(int oldfd, int newfd)`
  - jak wyżej, ale najpierw zamyka `newfd` i do niego kopiuje wpis odpowiadający `oldfd`
  - gubi błędy, które pojawiają się przy `close(newfd)`

Operacje różne:

- `fcntl`
  - `int fcntl(int fd, int cmd)`
  - `int fcntl(int fd, int cmd, long arg)`
  - `int fcntl(int fd, int cmd, struct flock *lock)`

```
while( (n = read(fd, buffer, BUFFER_SIZE) > 0 ) )  
{  
    write(fd2, buffer, n);  
}
```

Uwaga!

Wypadałoby obsługiwać błędy!

- Funkcja:  
`off_t lseek(int fd, off_t offset, int whence)`
- Parametry:
  - `fd` — deskryptor pliku, z którego następuje odczyt danych,
  - `offset` — wielkość przesunięcia (może być ujemne)
  - `whence` — odniesienie
    - `SEEK_SET`
    - `SEEK_END`
    - `SEEK_CUR`
- Wartość zwracana:
  - pozycja w pliku lub `-1` (*lub dowolna wartość*)



- Wg konwencji błędy oznaczane przez `return -1;`
- `errno`
  - wywołania systemowe
  - niektóre funkcje biblioteczne
  - zmienna wskazująca na numer błędu (lokalna dla wątku)
- `perror("We've got a problem")`
  - `We've got a problem: No such file or directory`

To work.