Laboratorium 7 – Obsługa potoków (łącza nienazwane)

Łącza w systemie LINUX są plikami specjalnymi, służącymi do komunikacji pomiędzy procesami. Łącza mają kilka typowych cech plików zwykłych, czyli posiadają swój *i*-węzeł, oraz bloki z danymi. Łącza od plików zwykłych odróżniają następujące cechy:

- ograniczona liczba bloków — mają rozmiar 4KB - 8KB w zależności od systemu,
- dostęp sekwencyjny — na łączach można wykonywać tylko operacje zapisu i odczytu, nie można wykonywać funkcji **`lseek`**,
- dane odczytywane z łącza są zarazem usuwane (nie można ich odczytać ponownie),
- proces jest blokowany w funkcji **`read`** na pustym łączu i w funkcji **`write`**, jeśli w łączu nie ma wystarczającej ilości wolnego miejsca, żeby zmieścić zapisywany blok (wyjątkiem od tej zasady jest przypadek, gdy jest ustawiona flaga `O_NDELAY`).

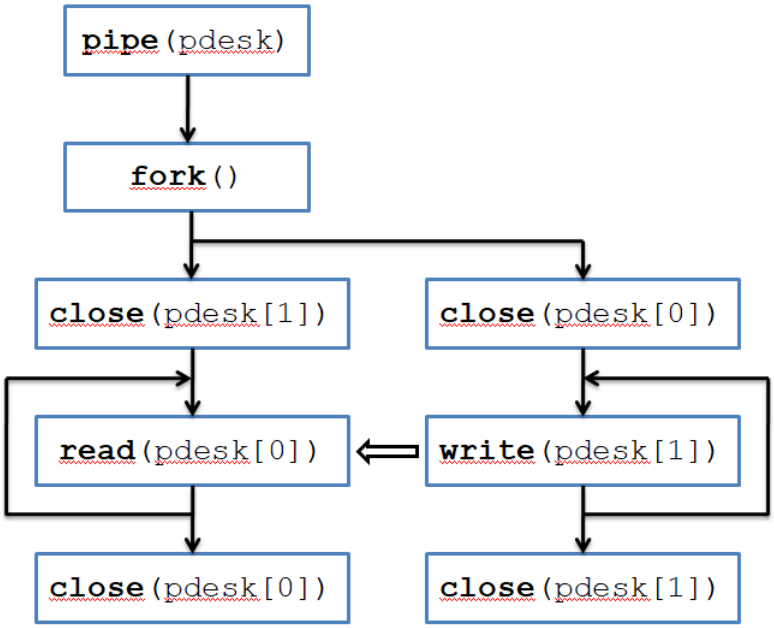
W systemie LINUX wyróżnia się dwa rodzaje łączy: łącza nazwane i łącza nienazwane. Zwyczajowo przyjęło się określać łącza nazwane terminem kolejki FIFO, a łącza nienazwane terminem potoki. Różnica pomiędzy łączem nazwanym i nienazwanym polega na tym, że łącze nazwane ma dowiązanie w systemie plików, a nienazwane takiego dowiązania nie posiada, czyli istnieje tak długo, jak długo jest otwarte. Po zamknięciu wszystkich deskryptorów łącze nienazwane przestaje istnieć i zwalniany jest jego *i*-węzeł oraz wszystkie bloki. Łącze nazwane natomiast po zamknięciu wszystkich deskryptorów w dalszym ciągu ma przydzielony *i*-węzeł, zwalniane są tylko bloki dyskowe. Jeżeli dwa procesy mają odpowiednie deskryptory łącza, to dla komunikacji między nimi nie ma znaczenia, czy są to deskryptory łącza nazwanego czy nienazwanego. Różnica występuje natomiast w sposobie uzyskania deskryptorów łącza. Potok tworzony jest w wyniku wywołania funkcji systemowej **`pipe`**.

`int` **`pipe`** `( int pdesk[2] )` – tworzy potok zwraca wartość 0 jeśli poprawnie zostało utworzone łącze nienazwane, w przeciwnym wypadku zwraca -1. Funkcja zdefiniowana w pliku `unistd.h`.

`pdesk[0]` – deskryptor potoku do odczytu,

`pdesk[1]` – deskryptor potoku do zapisu.

Komunikacja przez łącze wymaga, aby dwa procesy znały deskryptory tego samego łącza. Zatem proces, który utworzył potok może się przez niego komunikować tylko ze swoimi potomkami (niekoniecznie bezpośrednimi) lub przekazać im odpowiednie deskryptory, umożliwiając w ten sposób wzajemną komunikację (rys. 1).



Rys. 1. Schemat komunikacji przez potok.

Dane z potoku mogą być odczytane tylko raz przez jeden proces. Przy próbie odczytu z pustego łącza proces będzie blokowany w funkcji **read** do momentu zamknięcia wszystkich deskryptorów do zapisu. Jeśli funkcja **read** zwróci wartość 0 to oznacza, że wszystkie deskryptory do zapisu są zamknięte, a łącze jest puste. Jeśli rozmiar bufora przekazany jako trzeci parametr jest większy niż liczba danych w łączu to funkcja **read** zwróci wszystkie dane z łącza.

Funkcja **write** zapisuje w potoku liczbę bajtów określoną w zmiennej `count`. Jeśli nie jest możliwe zapisanie bloku danych ze względu na brak miejsca w łączu, to proces jest blokowany w funkcji **write** tak długo aż pojawi się odpowiednia ilość wolnego miejsca, tzn. aż inny proces odczyta i tym samym usunie dane z łącza.

`int` **dup** `( inf old_fd )` – tworzy duplikat istniejącego łącza i przekazuje numer nowego deskryptora związanego z tym samym łączem, zwraca wartość 0 lub -1 w przypadku błędnego zakończenia.

`old_fd` – deskryptor potoku,

Funkcja **dup** zwraca dostępny deskryptor łącza o najniższym numerze. Ponieważ oba deskryptory wskazują na to samo łącze to jedyna korzyść z zastosowania tej funkcji jest taka, że deskryptory będą miały różne numery, co pozwala np. zmienić standardowe łącze do zapisu lub odczytu.