Systemy operacyjne

Sprawozdanie - laboratorium 7
„Komunikacja międzyprocesowa 2”

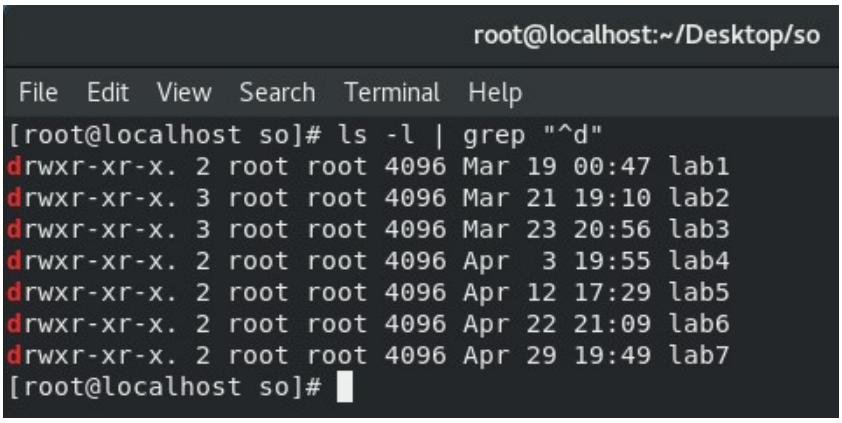
Andrzej Kołakowski
296586

1) Łącza nazwane w powłoce

Na poprzednim laboratorium poznaliśmy mechanizm komunikacji między procesami za pomocą łączy nienazwanych, tzw. potoków (ang. pipe).

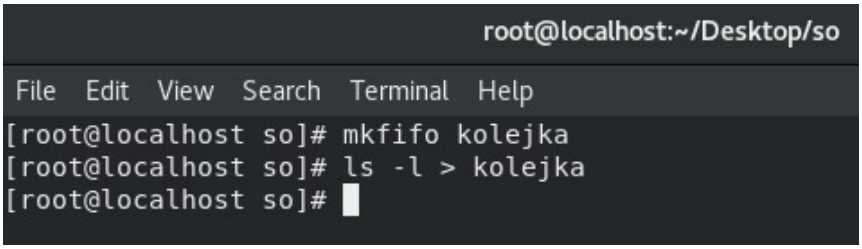
Łącze nazwane FIFO jest podobne do potoku, jednak jest tworzone w inny sposób. Nie jest anonimowym kanałem komunikacji, ale plikiem specjalnym dodawanym do systemu plików poprzez wywołanie `mkfifo`. Dzięki temu może być wykorzystane do komunikacji między dwoma procesami nieposiadającymi wspólnego przodka.

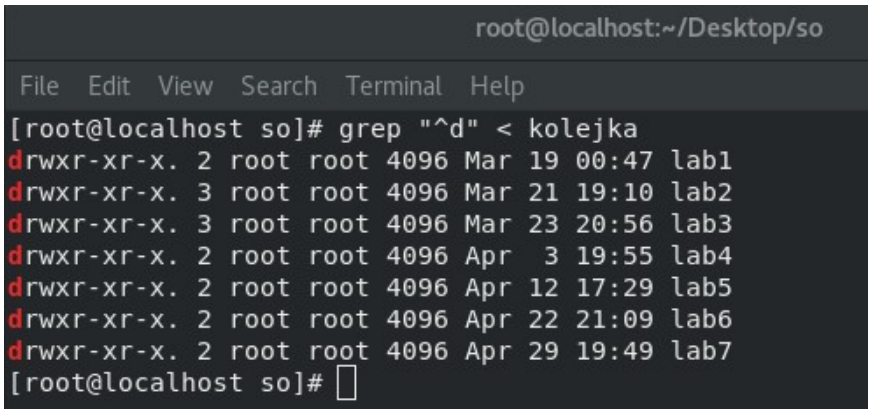
Aby wyświetlić same katalogi w bieżącym folderze można zastosować następujący filtr:



Analogiczny efekt można uzyskać za pomocą łącz nazwanych. Otwieramy dwie powłoki, aby nowotworzone procesy nie miały wspólnego przodka. Dzięki temu wykluczymy możliwość dziedziczenia deskryptorów.

W jednej z nich tworzymy kolejkę FIFO o dowolnej nazwie i uruchamiamy `ls -l` przekierowując jego standardowe wyjście na utworzoną kolejkę. W drugiej uruchamiamy `grep "^d"` przekierowując jego standardowe wejście na utworzoną kolejkę. Otrzymujemy analogiczny efekt.





2) Łącza nazwane w API

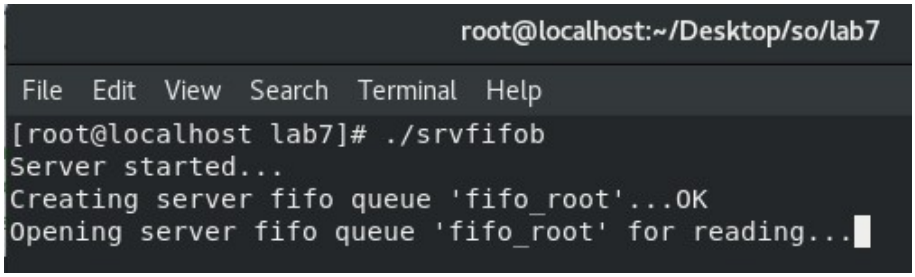
Rozważane programy implementują 3 schematy komunikacji klient-serwer między procesami wykorzystując łącza nazwane.

1. Programy wykorzystujące kolejki FIFO

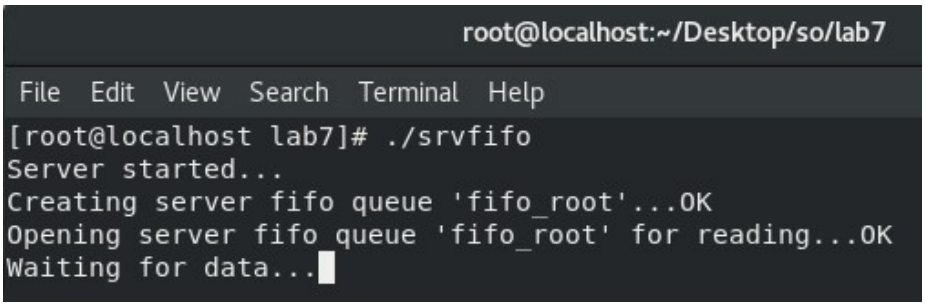
Wprowadzone modyfikacje:

- Serwer po uruchomieniu nie zatrzymuje się na otwieraniu łącza czytania danych od klienta, tylko przechodzi do oczekiwania na dane od klienta.
- Klient po wysłaniu komunikatu nie zatrzymuje się na otwieraniu łącza czytania danych od serwera, tylko przechodzi do oczekiwania na dane od serwera.

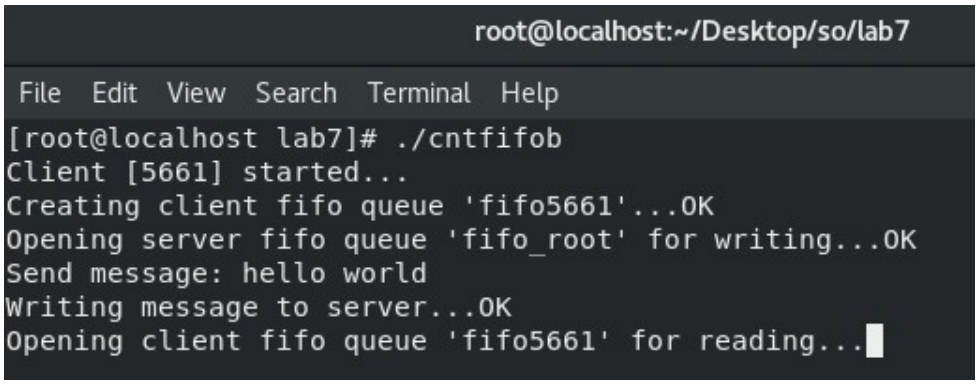
Domyślnie otwarcie kolejki FIFO w funkcji `open()` powoduje zatrzymanie wykonywania wątku do czasu aż nie zostanie otwarty przeciwległy koniec. Aby temu zapobiec należy dodać flagę `O_NONBLOCK`. Wtedy otwarcie do odczytu natychmiastowo zakończy się sukcesem, nawet jeżeli przeciwległy koniec nie został jeszcze otwarty. W funkcji `read()` chcemy jednak zatrzymać wykonanie wątku do czasu pojawienia się nadawcy, więc konieczne było usunięcie dodanej flagi. Dodatkowo należało zasymulować odbiorcę na drugim końcu – w przeciwnym wypadku `read()` natychmiast zwróciłoby **EOF** (w rozumieniu `read()` nadawca właśnie zamknął FIFO).



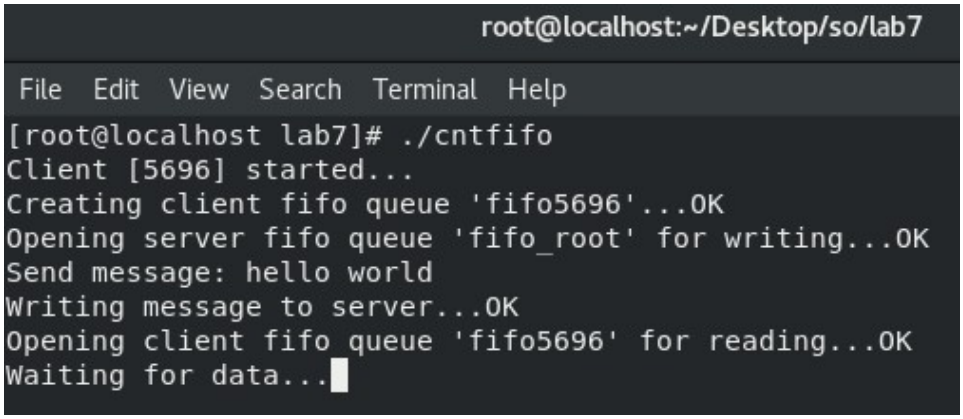
Serwer przed zmianami



Serwer po zmianach

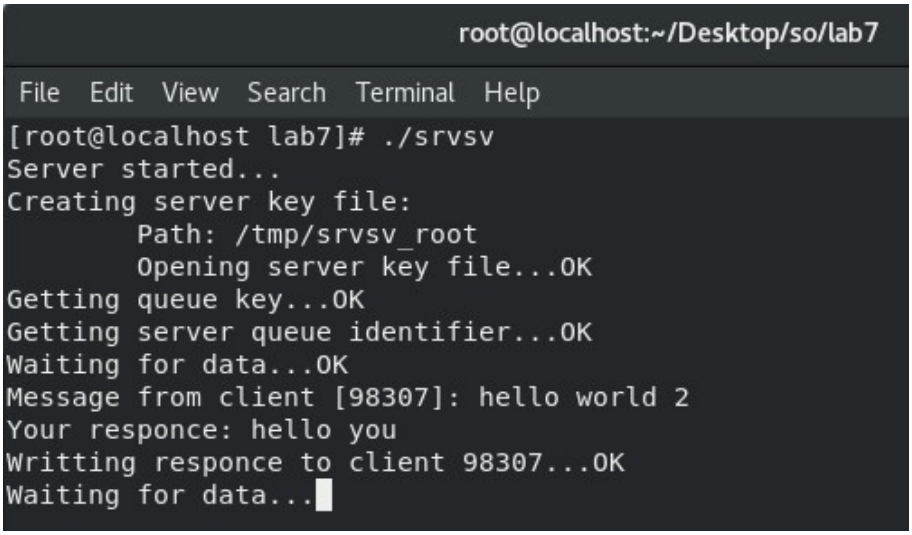


Klient przed zmianami

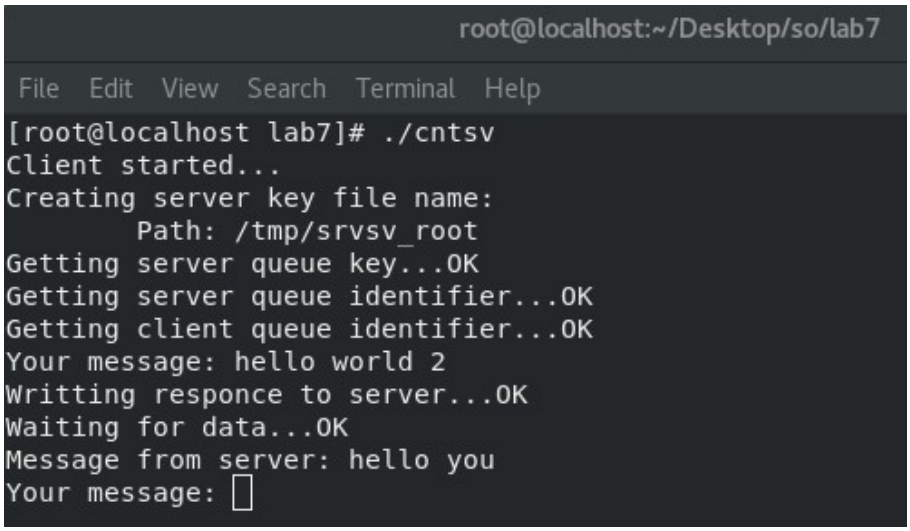


Klient po zmianach

2. Programy wykorzystujące interfejs z System V



Serwer

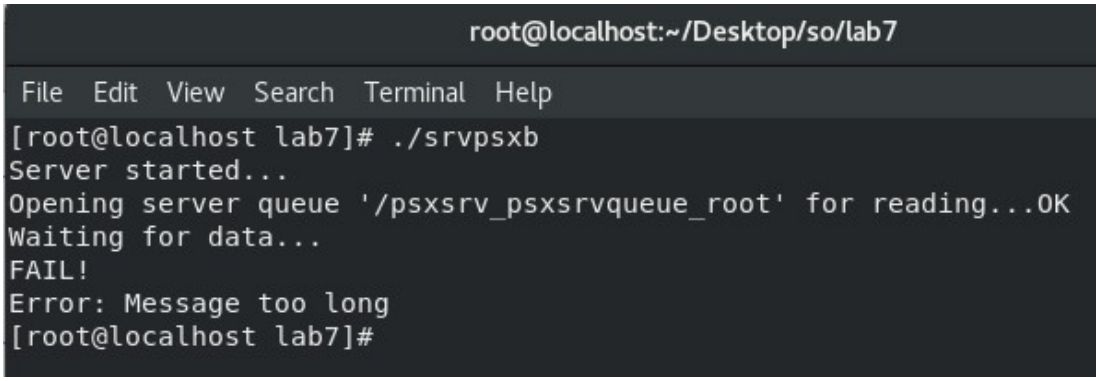


Klient

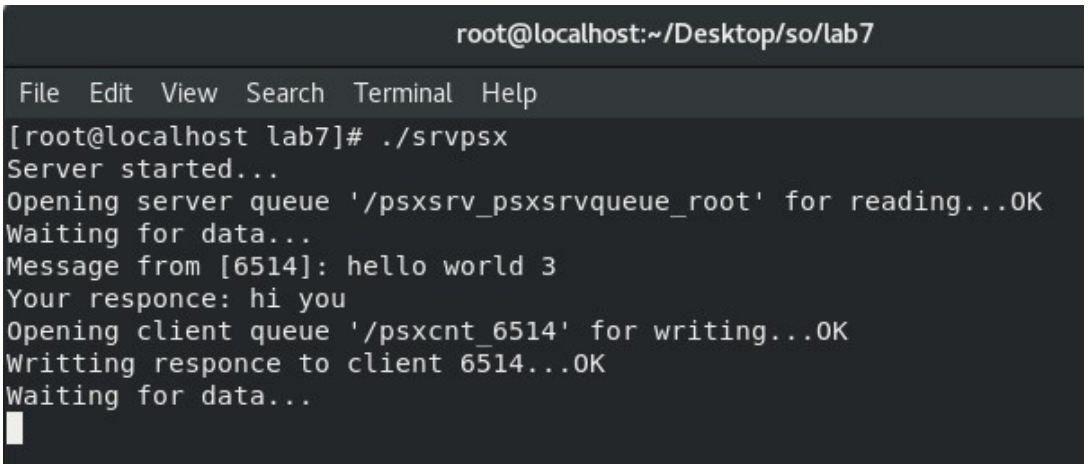
3. Programy wykorzystujące standard POSIX

Program zawierał błąd, a naszym zadaniem była jego identyfikacja. Po sprawdzeniu otrzymywanego `errno` okazuje się że dostajemy błąd `EMSGSIZE`, który oznacza że rozmiar bufora wiadomości jest mniejszy niż ten zawarty w atrybucie `mq_msgsize` utworzonej kolejki wiadomości. Tworząc kolejkę w funkcji `mq_open` jako `attr` podaliśmy **NULL**, co oznacza, że został użyty domyślny maksymalny rozmiar komunikatu. Można go odczytać za pomocą: `cat /proc/sys/fs/mqueue/msgsize_default`. U mnie wynosi on 8192 bajty. Zatem nasz bufor musi mieć rozmiar co najmniej 8192 bajtów. Po zmianie `MESSAGE_BUF_SIZE` w `psx.h` z 100 na 8192 i podaniu tej stałej do funkcji `mq_send` i `mq_receive` program zaczął działać poprawnie.

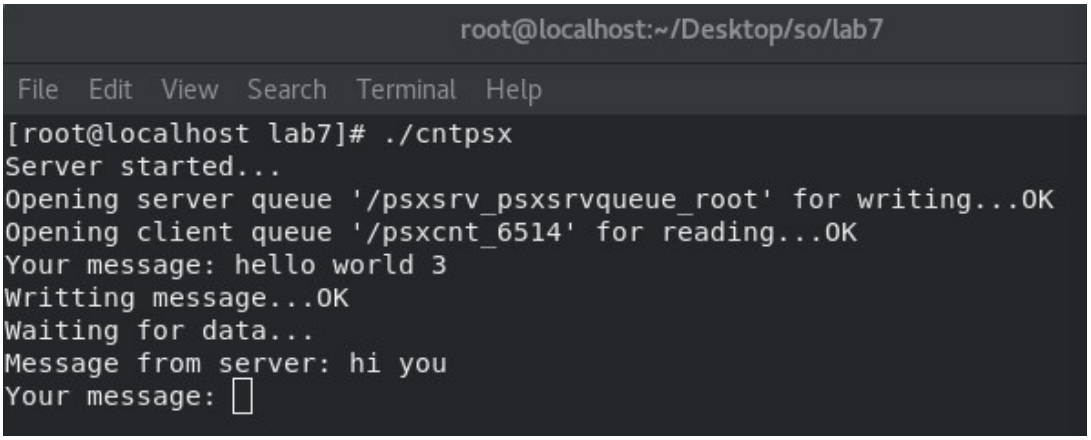
Maksymalną ilość wiadomości w kolejce można sprawdzić poleceniem: : `cat /proc/sys/fs/mqueue/msg_default`. U mnie wynosi ona 10.



Otrzymywany błąd



Serwer po zmianach

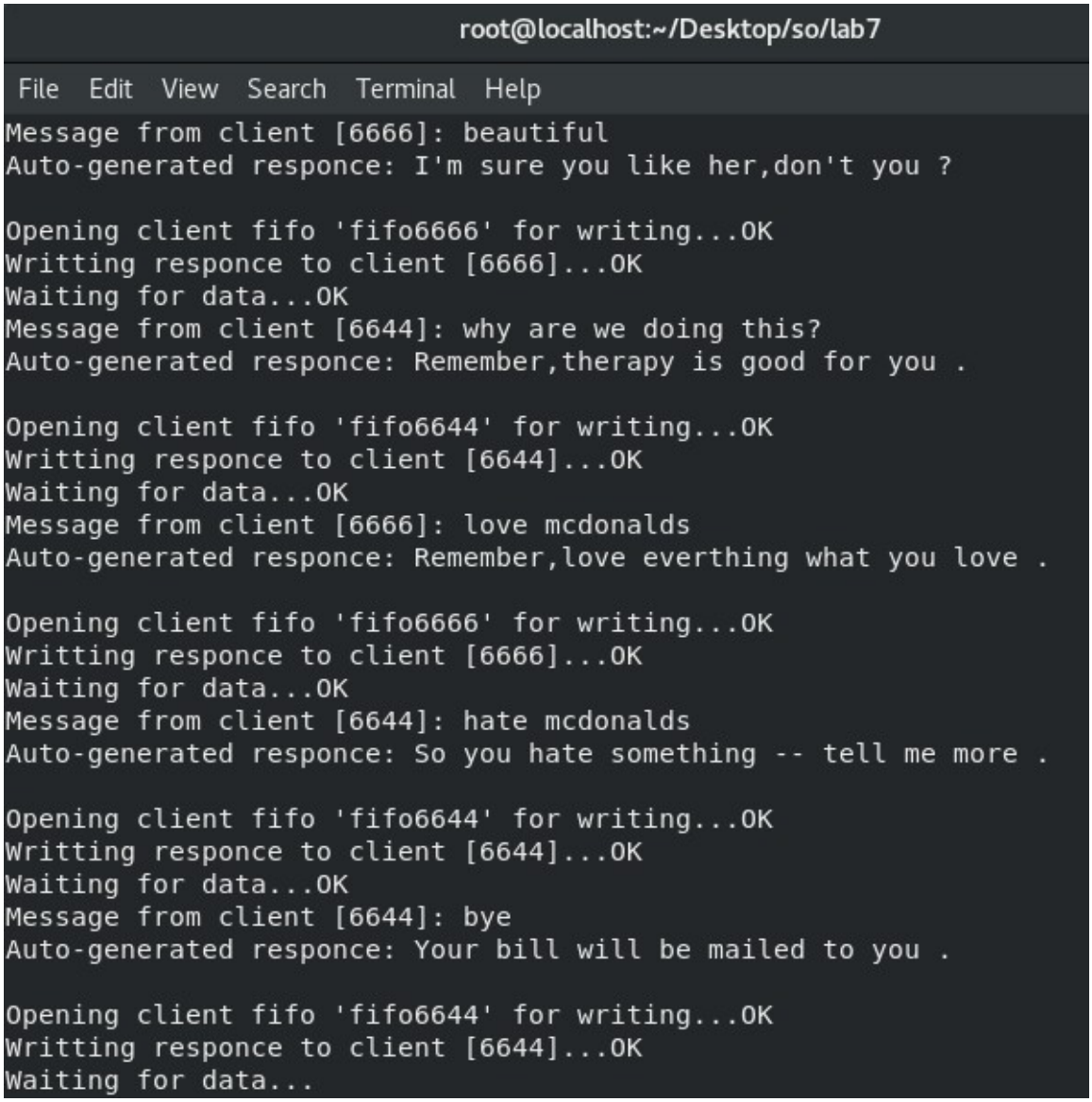


Klient po zmianach

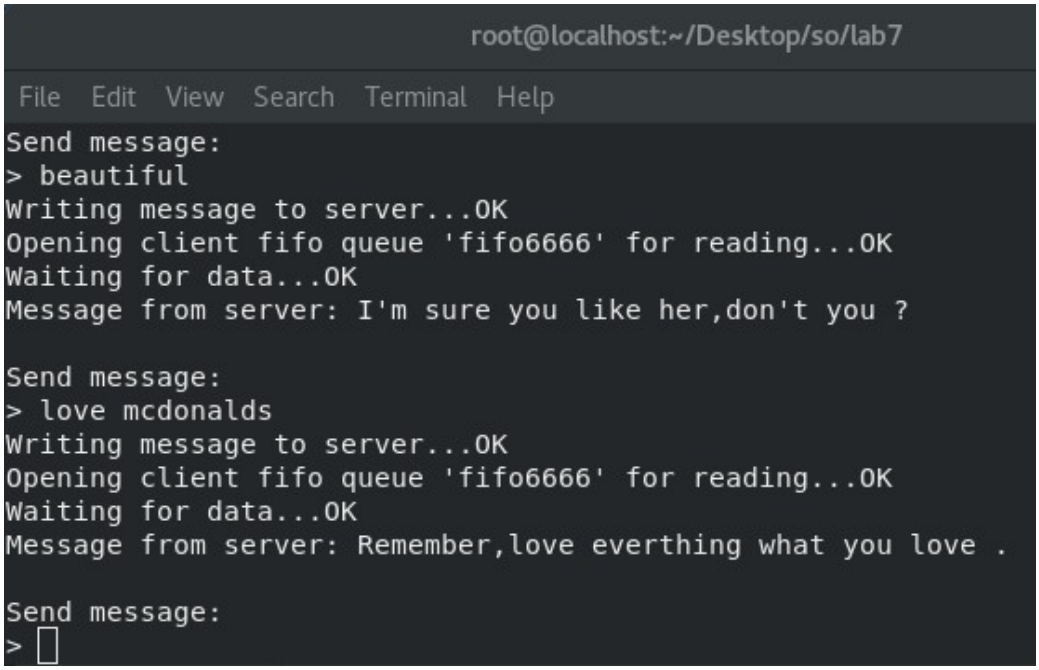
3) Zadania

1. Chatbot

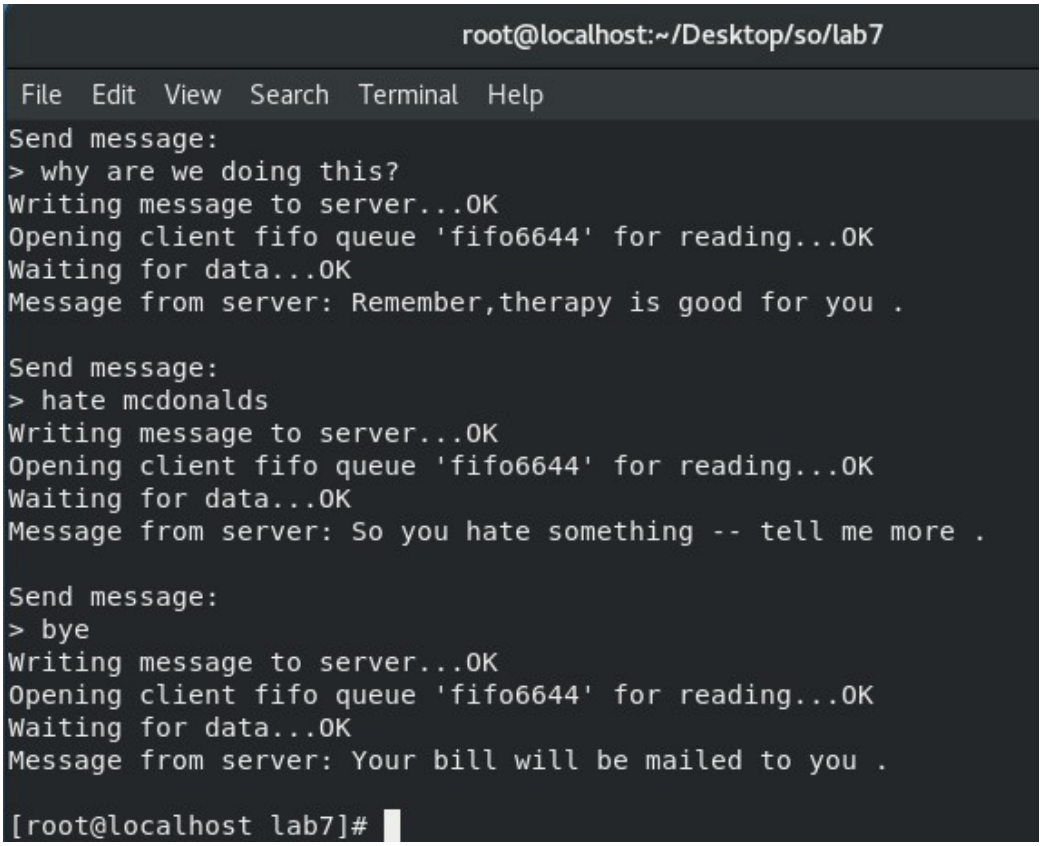
Zadanie polegało na modyfikacji chatbota Eliza tak, aby działał w trybie klient-serwer wykorzystując kolejki FIFO.



Serwer w trakcie rozmowy z dwoma klientami: 6666 i 6644



Klient 6666



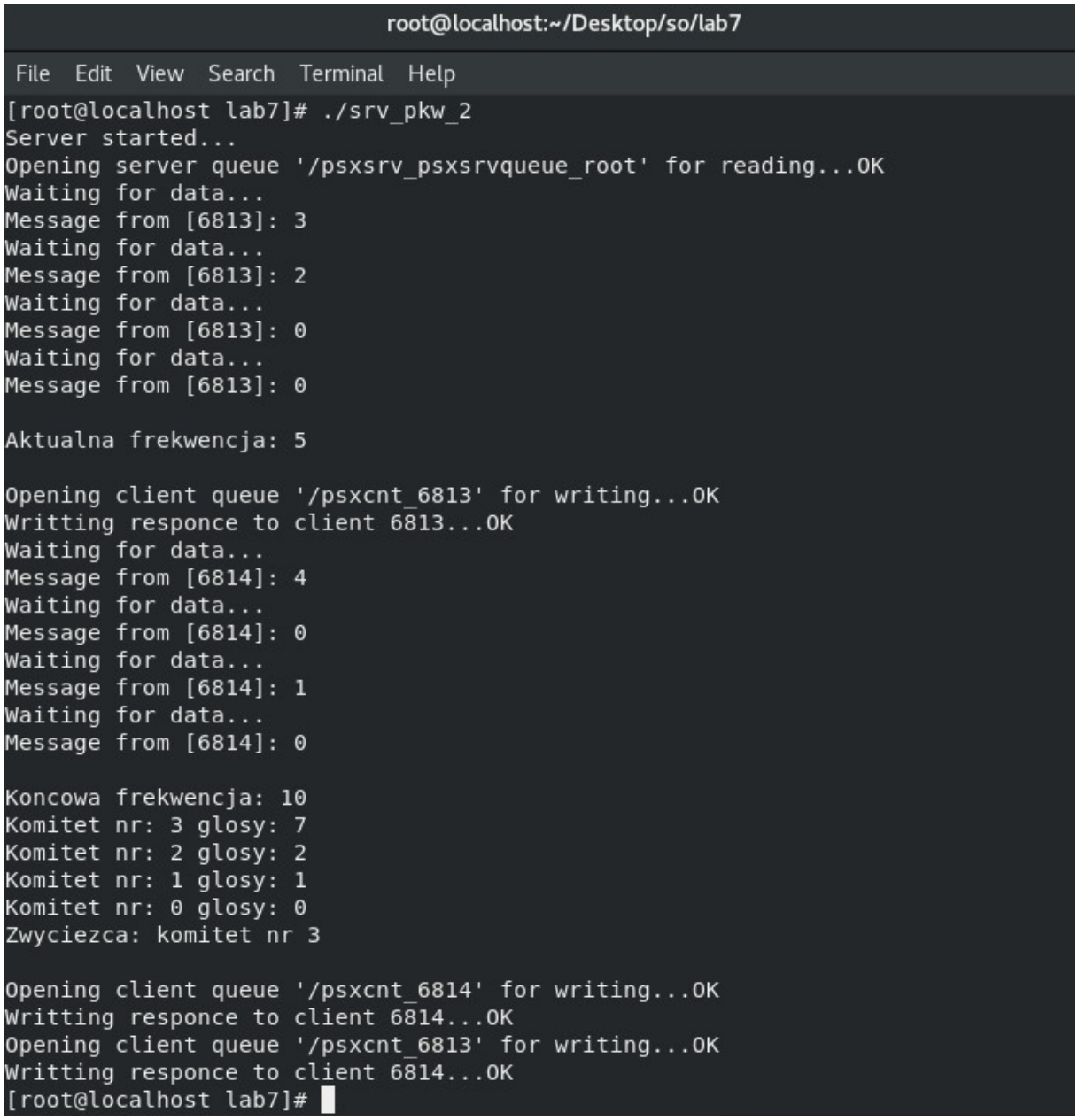
Klient 6644

2. Zliczanie głosów

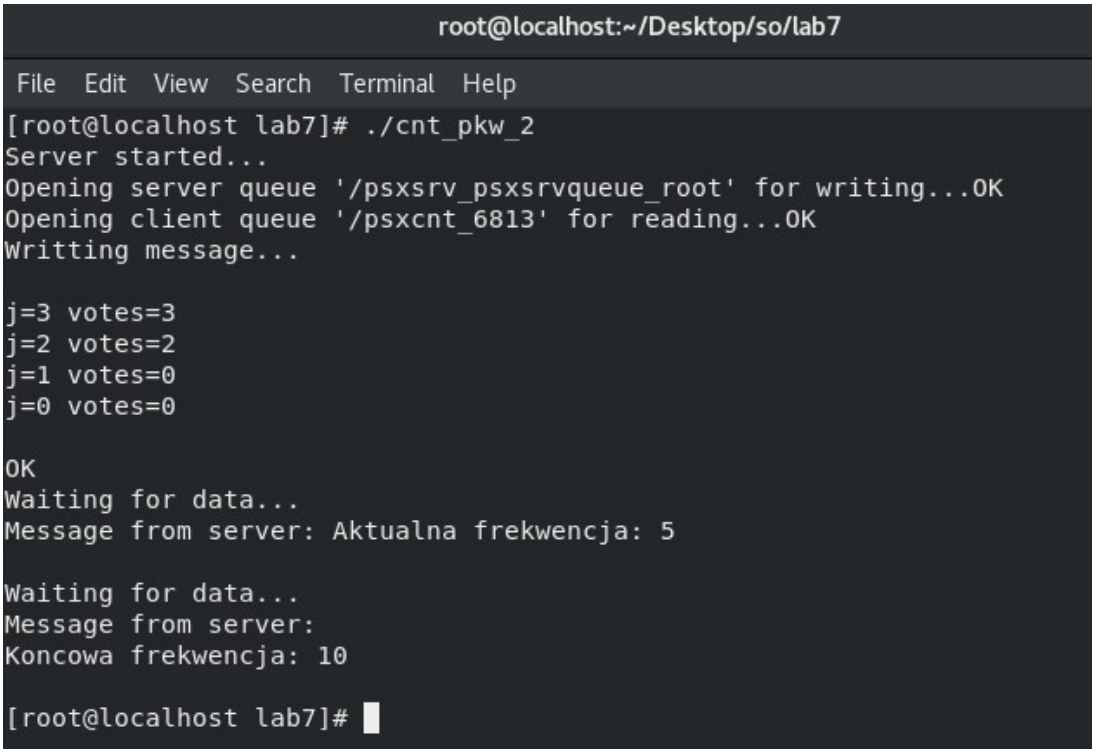
Zadanie polegało na napisaniu aplikacji, która zlicza głosy z poszczególnych okręgów wyborczych posługując się kolejką komunikatów. Do ich przesyłania wybrałem POSIX.

Założenia:

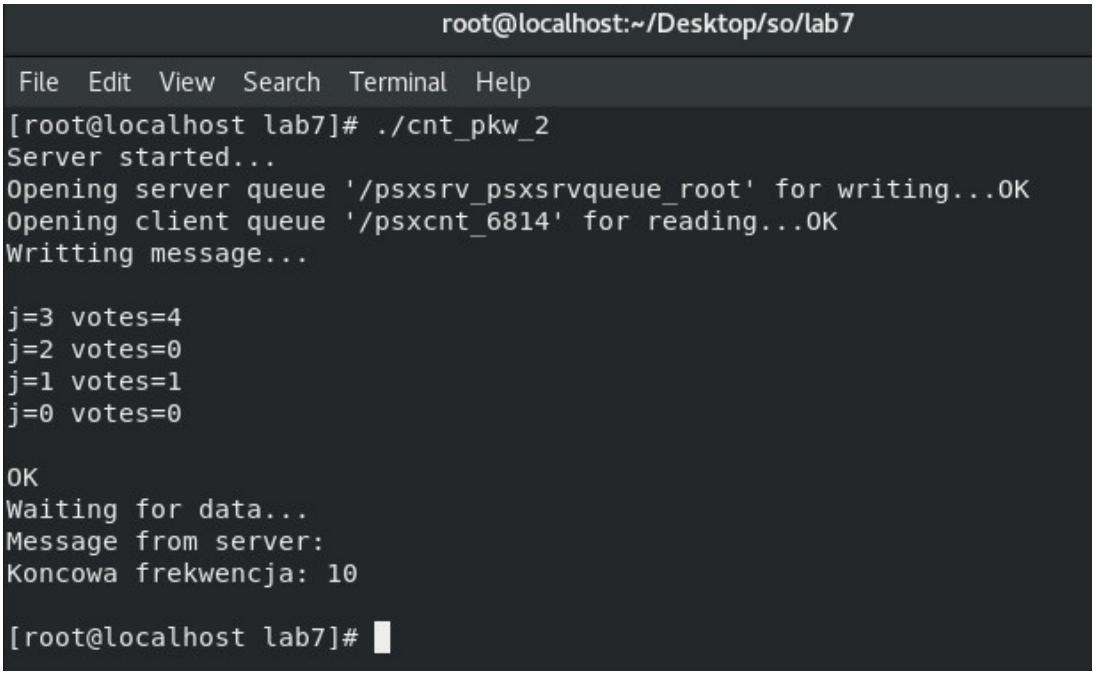
- Okręgi to aplikacje klienckie generujące losowo liczbę głosów
- Po każdorazowym otrzymaniu informacji od okręgu wyborczego, serwer przesyła do każdego zarejestrowanego okręgu aktualne dane o frekwencji wyborczej
- Informacje o liczbie głosów dla poszczególnych komitetów wyborczych są przesyłane poprzez priorytet komunikatu
- Na podstawie otrzymanych informacji serwer wyświetla informacje o frekwencji oraz który komitet wygrał wybory



Serwer odbierający dane z 2 okręgów wyborczych: 6813 i 6814



Klient (okręg) 6813



Klient (okręg) 6814