INSTYTUT TELEINFORMATYKI I AUTOMATYKI

Wydział Cybernetyki WAT

Przedmiot: SYSTEMY OPERACYJNE

SPRAWOZDANIE Z ĆWICZENIA LABORATORYJNEGO Nr 9

Temat ćwiczenia**: KOLEJKI KOMUNIKATÓW**

Wykonał:

Piotr Matyjek

Grupa: **I6Y3S1**

Ćwiczenie wykonane dnia
15.12.2017

Prowadzący ćwiczenie
mgr. inż. Stanisław Matusiak

Ocena:
……

Opis rozwiązania

Został mi przydzielony nr zadania 1. Programy zostały napisane z myślą, że najpierw jest uruchamiany jeden (lub więcej) odbiorca komunikatu, a następnie uruchamiany jest nadawca komunikatu. Oczywiście podczas pracy tych programów, można uruchomić kolejnych odbiorców. Kolejna idea przyświecająca mi podczas pisania tych programów, była taka, że to użytkownik uruchamiający programy musi decydować do którego programu odbierającego, wysyłany jest komunikat. Dlatego przy uruchomieniu każdego programu odbiorcy wyświetlany jest klucz, który go utworzył, jak i również pid tego procesu. Również program nadawcy, po odczytaniu ze standardowego wejścia linijki tekstu, czeka aby użytkownik podał pid procesu odbiorcy. Jeżeli użytkownik poda pid, który nie jest identyfikatorem żadnego z procesów-odbiorców, to komunikat będzie czekał w kolejce komunikatów aż do usunięcia kolejki. Ten komunikat nigdy nie zostanie odebrany przez żaden proces.

W programie matyjek_msgodb.c:

Na początku zdefiniowana została struktura komunikatu. Pierwszym jej składnikiem jest typ komunikatu, który jest typu „long". Następnie tablica znaków o długości 100. I ostatnim elementem jest liczba całkowita przez którą przesłany zostanie pid adresata. Ten element komunikatu został dodany zgodnie z poleceniem, lecz filtrowanie kolejki zostało wykonane bez użycia tej liczby całkowitej. Zostanie to omówione w dalszej części sprawozdania. Następnie przy użyciu funkcji „ftok" generuję klucz, który będzie wykorzystywany przy tworzeniu i używaniu kolejki komunikatów. Następnie przy użyciu funkcji „msgget" tworzę kolejkę komunikatów. Funkcja ta zwraca identyfikator kolejki, używany przy wysyłaniu i odbieraniu komunikatów. Następnie wyświetlany jest klucz, wygenerowany przez „ftok" i pid tego procesu, aby użytkownik mógł go podać procesowi nadawcy. Następnie w pętli „do while", proces wyczekuje komunikatów dla niego. Funkcja odpowiedzialna za odbiór komunikatów „msgrcv" jako czwarty argument przyjmuje typ odbieranego komunikatu. W moim programie ta funkcja będzie wyczekiwała komunikatu, którego typ jest zgodny z pid tego procesu (typ_komunikatu=pid). Jeśli te liczby nie będą równe, to funkcja najzwyczajniej zignoruje ten komunikat i będzie nadal wyczekiwała komunikatu. Jak liczby będą równe, to funkcja odbierze komunikat zapisując go w instancji „kom". Następnie wypisze swoje pid i treść komunikatu. Niestety element przechowujący pid procesu w komunikacie nie jest używany. Pętla „do while" będzie wykonywała się dopóki pierwszy znak w tablicy znaków przesyłanych przez kolejkę, będzie różny od '@' i dopóki funkcja „msgrcv" nie zwróci liczby „-1", co oznacza błąd w odbiorze komunikatu. W tym przypadku taki błąd występuje właśnie gdy użytkownik poda jako pierwszy znak linijki tekstu '@'. Wtedy komunikat jest wysyłany do odpowiedniego procesu-odbiorcy. Ten go odczytuje, wychodzi z pętli i kończy program. Jeśli był uruchomiony inny proces-odbiorcy to wtedy, w tym procesie „msgrcv" zwróci „-1". Ten warunek w pętli „do while", ma zapewnić żeby proce mógł się samoczynnie zakończyć po usunięciu kolejki.

W programie matyjek_msgnad.c:

Podobnie jak w programie odbiorcy, na początku definiowana jest struktura komunikatu. Następnie tworzone są potrzebne zmienne i argument przesłany podczas uruchomienia programu, zostaje za pomocą funkcji „sscanf" przekonwertowany z tablicy znaków, na liczbę całkowitą i przypisana ona zostaje do zmiennej „NR_kolejki". Następnie otwierana jest kolejka w tym procesie i program „wchodzi" w pętle „do while". Program prosi o podanie linijki tekstu, a następnie o podanie pid procesu do którego ma być wysłany komunikat. Pętla „while" opatrzona komentarzem „//bardzo wazna czesc kody, nigdy przenigdy nie wyrzucaj tego" została wstawiona, jako czyściciel bufora stdin. Następnie do odpowiednich elementów instancji struktury komunikatu „kom", zostają przypisane odpowiednie wartości. Komunikat zostaje wysłany. Gdy zmienna cont przyjmie wartość mniejszą od zera, będzie to oznaczało błąd wysłania komunikatu. Warunkiem wyjścia z pętli jest podanie jako pierwszy znak wpisywanego tekstu znaku '@'. Wtedy pętla się kończy, kolejka jest niszczona i proces zostaje zakończony.

Kody źródłowe programów

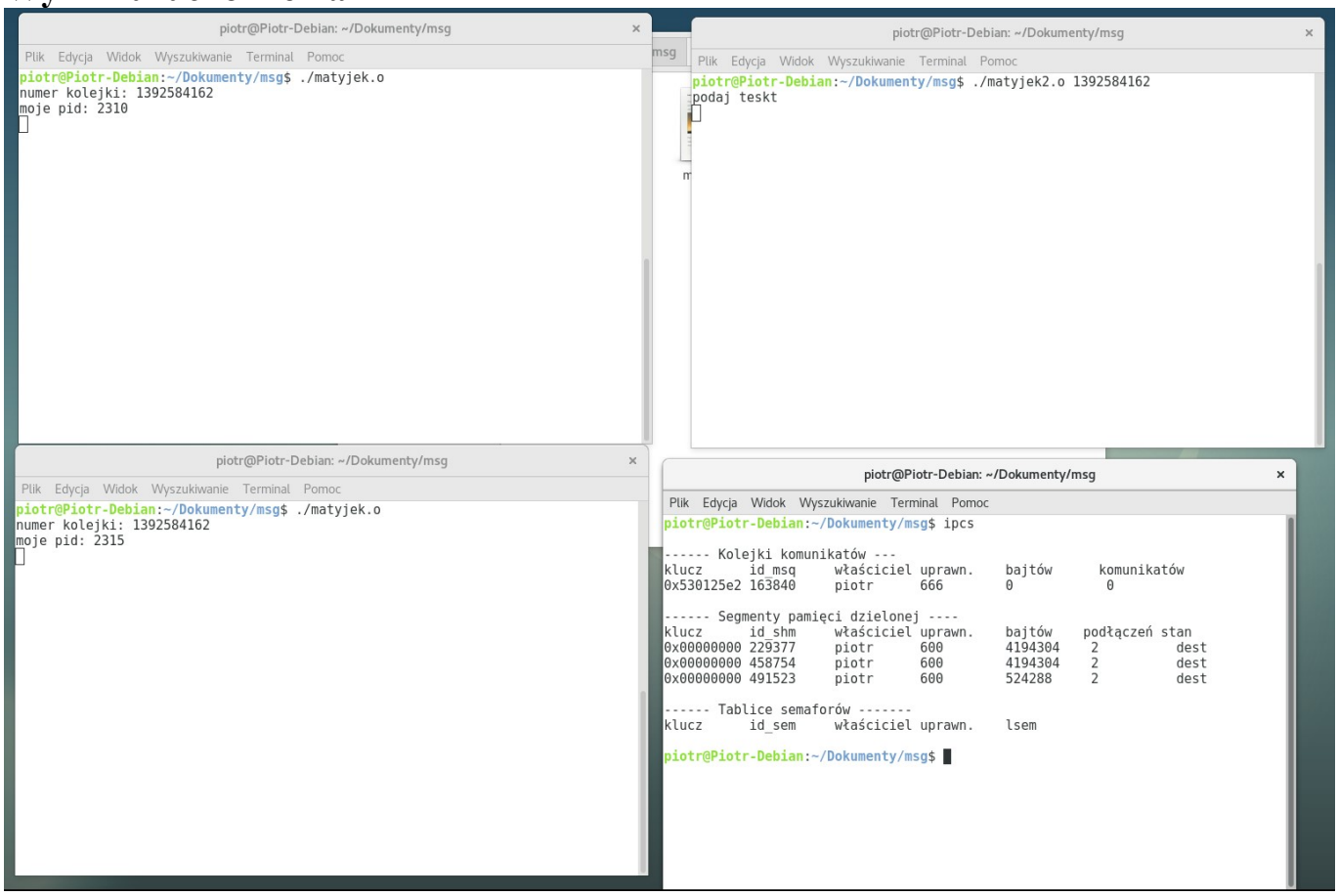
Matyjek_msgodb.c

```c
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
struct komunikat
{
long typ;
char linijka[100];
int pid;
};


int main()
{
key_t NR_kolejki;
int Kanal_WE, dlugosc, moje_pid=getpid();
//struct msqid_ds buf_WE;
struct komunikat kom;

NR_kolejki = ftok(".", 'S');

Kanal_WE=msgget(NR_kolejki, IPC_CREAT|0666);

printf("numer kolejki: %d\n",NR_kolejki);
printf("moje pid: %d\n",getpid());
do
{
dlugosc=msgrcv(Kanal_WE, &kom, 100+sizeof(int), moje_pid, 0);
printf("%d\t",moje_pid);

printf("%s\n",kom.linijka);


}while(kom.linijka[0]!='@' && dlugosc>0);

//msgctl(Kanal_WE, IPC_RMID, &buf_WE);
return 0;
}
```

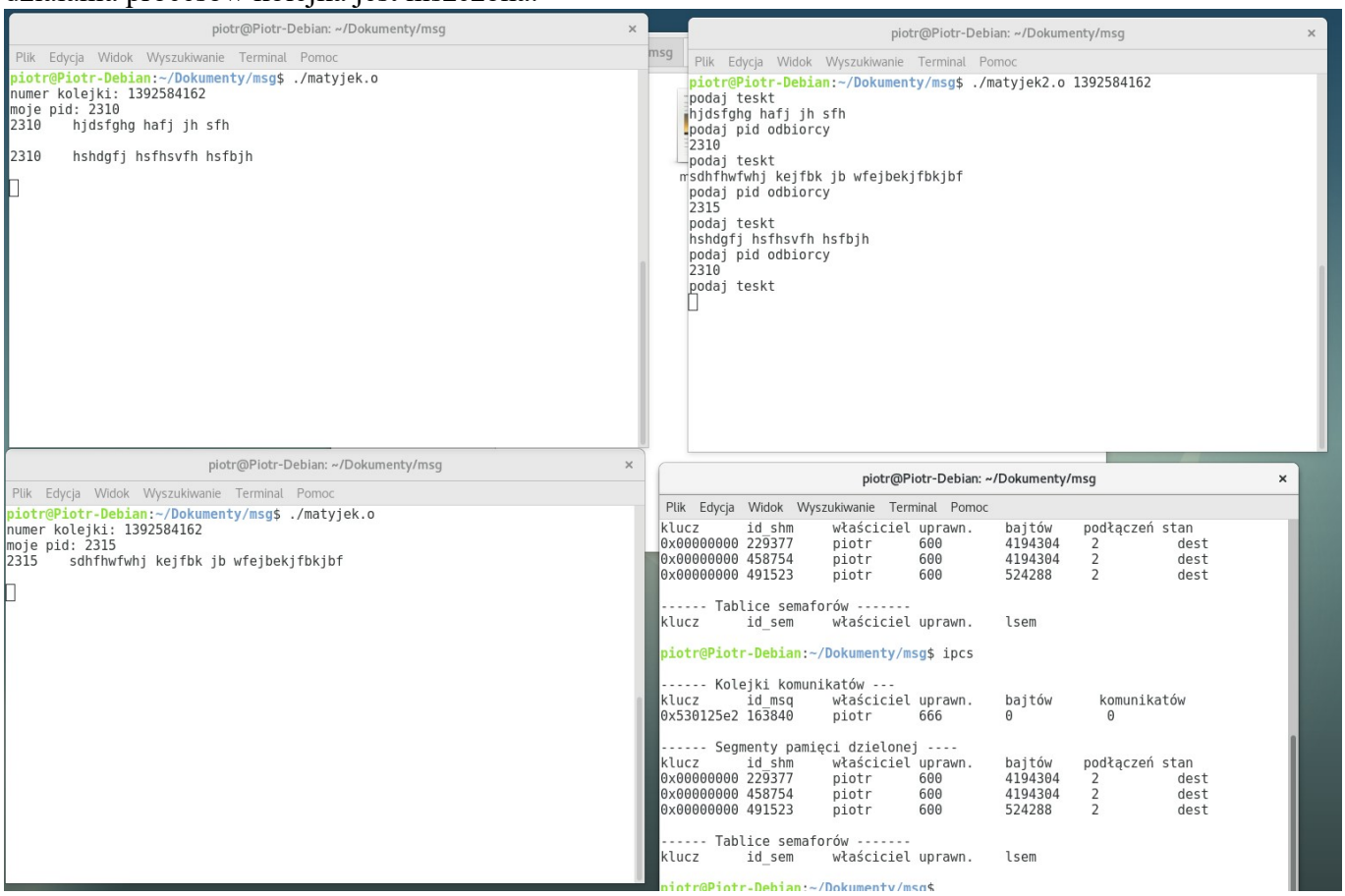
Matyjek_msgnad.c

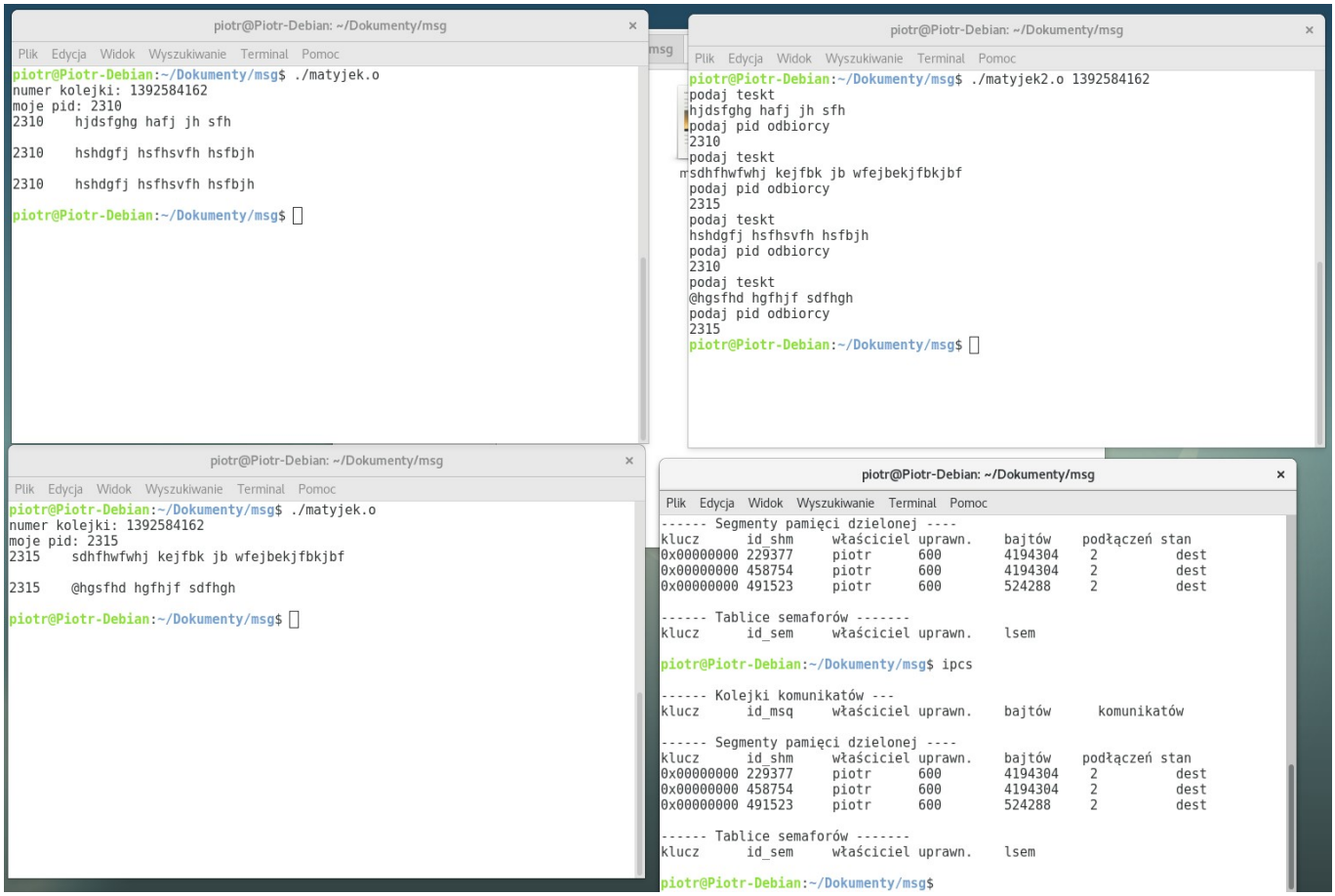
```c
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
struct komunikat
{
long typ;
char linijka[100];
int pid;
};

int main(int argc, char *argv[])
{
key_t NR_kolejki;
int Kanal_WY, dlugosc, pid_adr, cont;
struct msqid_ds buf_WY;
struct komunikat kom;
char tekst[100], c;

sscanf(argv[1], "%d", &NR_kolejki);
Kanal_WY=msgget(NR_kolejki, IPC_CREAT|0666);
do
{
        printf("podaj teskt\n");
        //printf("%d\n", NR_kolejki);
        fgets(tekst, 100, stdin);
        printf("podaj pid odbiorcy\n");
        //printf("%s\n", tekst);
        scanf("%d",&pid_adr);
                //bardzo wazna czesc kodu. nigdy przenigdy nie wyrzucaj tego!!!!!!
        while((c = getchar())!= '\n' && c != EOF)
        {
        printf("bla\n");
        }



        kom.typ = pid_adr;
        kom.pid=pid_adr;
        strncpy(kom.linijka, tekst, 100);


        cont=msgsnd(Kanal_WY, &kom, 100+sizeof(int),0);

        if(cont<0)
        {
        printf("blad wyslania komunikatu\n");
        return 2;
        }
}while(tekst[0]!='@');
msgctl(Kanal_WY, IPC_RMID, &buf_WY);
return 0;
}
```

Wyniki uruchomienia



Tuż po uruchomieniu. Dwie lewe konsole to dwa procesy-odbiorcy. Prawa górna konsola to proces-nadawca. A konsola prawa-dolna ma pokazać, że kolejka komunikatów jest utworzona i po zakończeniu działania procesów kolejka jest niszczona.





PODSUMOWANIE

Programy zostały napisane w języku C i uruchomione w środowisku systemu Linux Debian ver. 9. Programy wykonują się poprawnie, jak widać na zamieszczonych powyżej przykładach wywołania programów. Programy-odbiorcy powtarzają ostatnią odebraną linijkę po tym jak zostaje wysłany komunikat, który w tablicy znaków jako pierwszy znak ma ‘@’. Jest to spowodowane tym, że została zastosowana pętla „do while” w celu skrócenia długości kodu i zwiększenia jego czytelności. Program został napisany na zajęciach i to rozwiązanie było najszybsze i najefektywniejsze w tamtym czasie. Przesyłam w załączniku do tego sprawozdania pliki tych programów, ponieważ w domu poczyniłem drobne korekty kodów na przykład: usunięcie nieużywanych zmiennych w programie.