INSTYTUT TELEINFORMATYKI I AUTOMATYKI

Wydział Cybernetyki WAT

Przedmiot: SYSTEMY OPERACYJNE

SPRAWOZDANIE Z ĆWICZENIA LABORATORYJNEGO Nr 7

Temat ćwiczenia**: SYGNAŁY**

Wykonał:

Piotr Matyjek

Grupa: **I6Y3S1**

Ćwiczenie wykonane dnia
24.11.2017

Prowadzący ćwiczenie
mgr. inż. Stanisław Matusiak

Ocena:

……

Opis rozwiązania

Przydzielony został mi numer zadania 1. Kod programu realizującego zadanie został zamieszczony poniżej. Tuż pod załączeniami bibliotek, deklaruje dwie zmienne globalne odpowiadające za wykonywanie się pętli, w których procesy są nieaktywne. „sen1" jest dla procesu macierzystego, a „sen2" dla procesu potomnego. Procedura „obsluga", jest to procedura wywoływana w procesie potomnym, gdy ten otrzyma sygnał „SIGUSR1". Otwiera ona deskryptor pliku „bufor.txt", czyta całą linijkę tekstu w pętli while za pomocą funkcji „read" i jednocześnie wyświetla pojedyncze odczytane znaki na standardowe wyjście. Po zakończeniu pętli zamyka deskryptor i ustawia zmienną sen2 na 0. Następna procedura „obslugaUsun" wywoływana jest gdy proces potomny otrzyma sygnał „SIGUSR2". Powoduje ona zerwanie połączenia między „bufor.txt" a systemem plików, co efektywnie usuwa ten plik. Po usunięciu piku, proces wykonuje „exit(0)", który kończy jego pracę z kodem 0. Kolejna procedura „obsluga2" powoduje tylko zmianę zmiennej „sen1" na 0. Ta procedura wywoływana jest w procesie macierzystym, w przypadku otrzymania sygnału „SIGUSR1". Procedura „unstop" jest wywoływana w obu procesach, w przypadku napotkania sygnału „SIGSTOP". Powoduje ona tylko wysłanie do procesu który ją wywołał sygnału „SIGCONT". Obsługa „SIGSTOP" została wprowadzona z powodu podejrzenia, że system wysyła do wykonywanych procesów sygnał „SIGSTOP", przez co w pewnych momentach poprzednia wersja programu nie wykonywała się poprawnie. Nie dało to zamierzonego skutku, ale obsługa „SIGSTOP" pozostała na wszelki wypadek. Procedura „obslugaNIC" wywoływana gdyby do procesu macierzystego został wysłany sygnał „SIGUSR2", powoduje tylko wysłanie do procesu wywołującego sygnał „SIGCONT". Procedura podobnie jak poprzednia, została wprowadzona w poprzedniej wersji programu w celu weryfikacji, czy przypadkiem nie zostaje wysłany sygnał „SIGUSR2" do procesu macierzystego.

Na początku funkcji „main" deklarowane są zmienne m.in. dwa zestawy sygnałów, które będą wykorzystane jako maski w procesie potomnym i macierzystym. Po wykonaniu funkcji „fork", w procesie potomnym inicjalizowana jest maska sygnałów, wprowadzana jest obsługa sygnałów, zmieniana jest wartość zmiennej „sen2" na 1. Następnie proces wchodzi w wykonywanie pętli nieskończonej, w której zagnieżdżona jest pętla wykonująca się gdy zmienna „sen2" jest różna od 0, a w niej funkcja „pause" zatrzymuje wykonywanie procesu, do czasu aż jakiś sygnał dotrze do tego procesu. Jednak tylko sygnał „SIGUSR1", pozwoli na wyjście z tej pętli.

Gdy proces otrzyma wyżej wymieniony sygnał i pętla zostanie zakończona, proces zostaje jeszcze uśpiony na dziesięć milisekund (10 000 μ sekund), aby po otrzymaniu sygnału i wykonaniu obsługi tego sygnału proces mógł definitywnie zakończyć czynności związane z czytaniem deskryptora i wypisywaniem na ekran linijek tekstu. Czyli funkcja „usleep" jest, aby zniwelować występowanie hazardów. Po „wybudzeniu" się procesu, ten wysyła do procesu macierzystego sygnał „SIGUSR1", ustawia „sen2" na wartość 1, po czym wykonywanie się procesu zapętla się i znów wykonywana jest pętla „while" z warunkiem „sen2". Proces będzie wykonywał pętle nieskończoną „while(1)", aż do otrzymania „SIGUSR2".

Na początku wykonywania procesu macierzystego, inicjalizowana jest maska sygnałów i obsługa sygnałów. Następnie otwierany jest plik „/etc/profile", przy użyciu wskaźnika do pliku „FILE *fp", który został zadeklarowany na początku funkcji main. Kolejny plik otwierany jest przy użyciu deskryptora pliku. Jest to „bufor.txt". Otwierany jest przy użyciu funkcji „open" z argumentami: „O_WRONLY" – tylko zapis, „O_CREAT"-gdy plik nie istnieje, to go utwórz, „O_TRUNC"-po otwarciu nadpisuj plik, „S_IRWXU"-nadaje użytkownikowi prawa czytania, zapisu i wykonywania. Proces zaczyna wykonywać pętle z warunkiem czytania z pliku „/etc/profile". Następnie wpisuje wczytany pojedynczy znak do „bufor.txt" przy użyciu funkcji „write". Następnie sprawdza, czy przeczytany znak z „/etc/profile" jest znakiem, końca linii. Jeżeli nie, to przechodzi do kolejnego wykonywania pętli. Jeżeli znak był znakiem końca linii, to proces zamyka deskryptor pliku, wysyła sygnał do procesu potomnego „SIGUSR1", zmienia zmienną „sen1" na wartość 1 i wchodzi do pętli „while" z warunkiem „sen1", w której znajduje się funkcja „pause". Proces ten nie wyjdzie z tej pętli dopóki nie otrzyma od procesu potomnego sygnału „SIGUSR1". Po otrzymaniu tego sygnału, wykonywana jest procedura obsługi sygnału, czyli zmiana „sen1" na wartość 0, co spowoduje wyjście z tej pętli. Następnie otwierany jest deskryptor pliku i proces zaczyna znów czytać „/etc/profile". Gdy „/etc/profile" zostanie cały przeczytany, to wtedy proces wyjdzie z zewnętrznej pętli i wyśle sygnał „SIGUSR2", do procesu potomnego, a ten usunie „bufor.txt".

Następnie proces macierzysty kończy pracę z „/etc/profile” przy użyciu „fclose” i sam kończy pracę używając „exit(0)”.

Kod źródłowy programu

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



int sen1=0;
int sen2=0;

void obsluga (int sig)
{
char c;
int filedesc;

                if((filedesc=open("bufor.txt",O_RDONLY))<0)
                {
                printf("error filedesc1-potomny\n");
                exit(2);
                }
                lseek(filedesc,0,SEEK_SET);


                while((read(filedesc, &c, 1))>0)
                {
                printf("%c",c);
                }

                if((close(filedesc))!=0)
                {
                printf("problem z zamknieciem filedesc (proces pot)\n");
                exit(2);
                }


                sen2=0;


}
void obslugaUsun(int sig)
{
        if(unlink("/home/piotr/Dokumenty/bufor.txt"))
        {
        printf("blad usuniecia pliku\n");
        exit(3);
        }
exit(0);

}

void obsluga2 (int sig)
{


sen1=0;

}
```

```
void unstop (int sig)
{
raise(SIGCONT);
}

void obslugaNIC (int sig)
{
raise(SIGCONT);
}

int main(void)
{
FILE *fp;

int pid, pidparent, filedesc;
sigset_t signalSet;
sigset_t signalSet2;



if((pid=fork())<0)
{
printf("cos nie wyszlo\n");
exit(1);

}
else
{
        if(pid==0)
        {

        //sigset_t *signalSet;
        if(sigemptyset(&signalSet))printf("error!\n");
        if(sigfillset(&signalSet))printf("errrrrrr!\n");
        if(sigdelset(&signalSet,SIGUSR1))printf("error!\n");
        if(sigdelset(&signalSet,SIGUSR2))printf("error!\n");
        sigdelset(&signalSet,SIGCONT);
        sigprocmask(SIG_SETMASK, &signalSet,NULL);

        pidparent=getppid();


        signal(SIGUSR1,obsluga);
        signal(SIGUSR2,obslugaUsun);
        signal(SIGCONT,obsluga);
        signal(SIGSTOP,unstop);

        sen2=1;

                while(1)
                {

                        while(sen2)
                        {
                        pause();
                        }
                        usleep(10000);


                        kill(pidparent,SIGUSR1);
                        sen2=1;
                }
```

```
}
else
{

//sigset_t *signalSet2;
if(sigemptyset(&signalSet2))printf("error!\n");
if(sigfillset(&signalSet2))printf("errrrrrr!\n");
if(sigdelset(&signalSet2,SIGUSR1))printf("error!\n");
        sigdelset(&signalSet2,SIGUSR2);
        sigdelset(&signalSet2,SIGCONT);

sigprocmask(SIG_SETMASK, &signalSet,NULL);
int sig1, i=0;
char c;


        signal(SIGUSR1,obsluga2);
        signal(SIGCONT,obsluga2);
        signal(SIGSTOP,unstop);
        signal(SIGUSR2,obslugaNIC);

        if((fp=fopen("/etc/profile","r"))==NULL)
        {
        printf("cos nie teges\n");
        exit(1);

        }
                if((filedesc=open("bufor.txt",O_WRONLY|O_CREAT|O_TRUNC,S_IRWXU))<0)
                {
                printf("error filedesc1\n");
                exit(1);
                }



        while(fscanf(fp,"%c",&c)>0)
        {



                if((write(filedesc, &c, 1))<0)
                {
                printf("problem z wpisaniem do bufora\n");
                exit(1);
                }
              if(c=='\n')
                     {

                                if((close(filedesc)))
                                {
                                printf("problem z zamknieciem filedesc (proces mac)\n");
                                exit(1);
                                }

                                if(kill(pid,SIGUSR1))printf("blad wyslania sygnalu\n");
                                sen1=1;



                                while(sen1)
                                {
                                pause();
                                }
```

```
                                        if((filedesc=open("bufor.txt",O_WRONLY|O_CREAT|O_TRUNC,S_IRWXU))<0)
                                        {
                                        printf("error filedesc1\n");
                                        exit(1);
                                        }

                                }


                }

                kill(pid,SIGUSR2);

        fclose(fp);

        }

}


exit(0);
}
```

Wynik uruchomienia

```
piotr@Piotr-Debian:~/Dokumenty$ gcc matyjek_sig3.c -o matyjek.o
piotr@Piotr-Debian:~/Dokumenty$ ./matyjek.o
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
fi
export PATH

if [ "${PS1-}" ]; then
  if [ "${BASH-}" ] && [ "$BASH" != "/bin/sh" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi
piotr@Piotr-Debian:~/Dokumenty$
```

PODSUMOWANIE

Program został napisany w języku C i uruchomiony w środowisku systemu Linux Debian ver. 9. Program wykonuje się poprawnie, jak widać na zamieszczonych powyżej przykładzie wywołania programu. Do tego sprawozdania dołączam plik z kodem programu, którego nie zdołałem wysłać na zajęciach. Jedyne zmiany jakie zostały dokonane w stosunku do zaprezentowanego na zajęciach kodu, to zmniejszenie odstępów między fragmentami kodu i zwiększenie czytelności, poprzez poustawianie niektórych linijek kodu z odpowiednimi odstępami tabulacjami.