

INSTYTUT TELEINFORMATYKI I AUTOMATYKI

Wydział Cybernetyki WAT

Przedmiot: SYSTEMY OPERACYJNE

SPRAWOZDANIE Z ĆWICZENIA LABORATORYJNEGO Nr 11

Temat ćwiczenia: MUTEXY W SYSTEMACH UNIX\LINUX

Wykonał:

Damian Krata

Grupa: I4X3S1

Ćwiczenie wykonane dnia
10.01.2016


Prowadzący ćwiczenie
mgr inż. Łukasz Laszko

Ocena:

.....

Treść zadania

1



Napisać **trójwatkowy** program do przesyłania wiadomości (wątek główny, czytający, piszący) z synchronizacją zrealizowaną za pomocą **muteksów** POSIX. Wątek główny powołuje dwa wątki: czytający i piszący. Wątek czytający i piszący powinny działać współbieżnie, wykorzystując do komunikacji **współdzielony bufor** o rozmiarze 8 bajtów.

Odpowiedź:

☐ Prawda

☐ Fałsz

Zatwierdź

Opis rozwiązania

W celu wykonania tego zadania, powołałem wątek główny, który z kolei powołuje dwa inne procesy, czytający i piszący. W moim projekcie końcowym mam podobne zadanie więc uzyskałem gotowego kodu, który wcześniej napisałem i pomniejszyłem go o obsługę sygnałów etc. Ponadto zmieniłem tylko wartość bufora na 8 bajtów, czyli będę mógł wczytywać jedną linię i 8 znakach. Co to są mutexy? Są to blokady, które może uzyskać tylko jeden wątek. Mutexy służą głównie do realizacji sekcji krytycznych, czyli bezpiecznego w sensie wielowątkowym dostępu do zasobów współdzielonych.

Schemat działania na mutexach jest następujący:

1. pozyskanie blokady
2. modyfikacja lub odczyt współdzielonego obiektu
3. zwolnienie blokady

Mutex w pthreads jest opisywany przez strukturę typu `pthread_mutex_t`, zaś jego atrybuty `pthread_mutexattr_t`.

Do rozwiązania tego zadania korzystałem również z amerykańskiego tutorialu o mutexach w wątkach

<https://www.youtube.com/watch?v=GXXE42bkqQk>

Kod Programu:

```
#include <sys/syscall.h>
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>

#define BUFFSIZE 1
#define BUFFSIZE2 1
#define LINESIZE 8

pid_t tid1;
pid_t tid2;
pthread_t watek1;
pthread_t watek2;

int pauza1, pauza2;
int koniec1, koniec2;

char buf [BUFFSIZE][LINESIZE];

pthread_mutex_t lockprint;

pthread_mutex_t lock;
pthread_cond_t notempty;
pthread_cond_t notfull;

int readposition, writeposition;

int itemCount; //ile jest linii w buforze
void* funkcja_watek1(void* dane);
void* funkcja_watek2(void* dane);

void printbuf()
{
    pthread_mutex_lock (&lockprint);
    int i;
    for (i=0;i<BUFFSIZE;i++)
    {
        printf ("%d %s\n", i , buf[i]);
    }
    pthread_mutex_unlock (&lockprint);
}

void wstaw_do_bufora(char* linia)
{
    pthread_mutex_lock (&lock); //blokuje bufor zeby wstawiac
    while(itemCount==BUFFSIZE)
    {
        pthread_cond_wait(&notfull,&lock);
```

```

    }

    strcpy(buf[writeposition],linia);
    printf("watek1 wstawil kolejna linie do bufora \n");

    printf();
    writeposition++;
    if(writeposition==BUFFSIZE)
        writeposition=0;

    itemCount++;
    pthread_cond_signal (&notempty);

    pthread_mutex_unlock (&lock);    //odblokowuje bufor
}

```

```

void pobierz_z_bufora(char* linia)
{
    pthread_mutex_lock (&lock); //blokuje bufor zeby wstawiac
    while(itemCount==0)
    {
        pthread_cond_wait(&notempty,&lock);
    }

    strcpy(linia,buf[readposition]);
    strcpy(buf[readposition],"-");
    printf("watek2 zabral kolejna linie z bufora \n");
    printf();
    readposition++;
    if(readposition==BUFFSIZE)
        readposition=0;

    itemCount--;
    pthread_cond_signal (&notfull);

    pthread_mutex_unlock (&lock);    //odblokowuje bufor
}

```

```

////////////////////////////////////

```

```

int buf2 [BUFFSIZE2];

```

```

pthread_mutex_t lock2;
pthread_cond_t notempty2;
pthread_cond_t notfull2;

```

```

int readposition2, writeposition2;

```

```

int itemCount2; //ile jest linii w buforze

```

```

void init()
{
    int i;
    for (i=0;i<BUFFSIZE;i++)
    {
        strcpy(buf[i], "-");
    }
}

```

```

    }
    pthread_mutex_init(&lock,0); //ustawiam na otwarty
    pthread_cond_init(&notempty,0);
    pthread_cond_init(&notfull,0);
    itemCount=0;
    readposition=0;
    writeposition=0;

    pthread_mutex_init(&lockprint,0);
    pauza1=0;
    pauza2=0;
    koniec1=0;
    koniec2=0;
}

void* mainwatek (void* z)
{
    printf("utworzyłem watek główny\n");
    pthread_t watekczytajacy;
    pthread_t watekpiszacy;
    pthread_create( &watek1, NULL, funkcja_watek1, NULL);
    pthread_create( &watek2, NULL, funkcja_watek2, NULL);
    pthread_join(watek1,NULL);
    pthread_join(watek2,NULL);
}

void* funkcja_watek1(void* dane)
{
    tid1 = (pid_t) syscall (SYS_gettid);
    pid_t pid=getpid();
    pid_t ppid=getppid();
    printf ("watek 1 rozpoczyna działanie TID: PID: PPID: %d %d %d \n",tid1,pid,ppid);
    char linia[LINESIZE];
    while ((fgets(linia,LINESIZE,stdin)!=NULL)&&(koniec1==0))
    {
        if (pauza1==0)
        {
            linia[strlen(linia)-1]='\0';
            printf("watek1 przeczytał z wejścia linie: %s\n", linia);
            wstaw_do_bufora(linia);
        }
    }

    printf("watek 1 kończy działanie\n");
}

void* funkcja_watek2(void* dane)
{
    tid2 = (pid_t) syscall (SYS_gettid);
    pid_t pid=getpid();
    pid_t ppid=getppid();
    printf ("watek 2 rozpoczyna działanie TID: PID: PPID: %d %d %d \n",tid2,pid,ppid);
    char linia[LINESIZE];
    while (koniec2==0)
    {

```

```

        pobierz_z_bufora(linia);
        printf("watek2 przeczytał z bufora linie: %s\n", linia);
        while(pauza2==1) {}
    }
    printf("watek 2 konczy dzialanie\n");
}

```

```

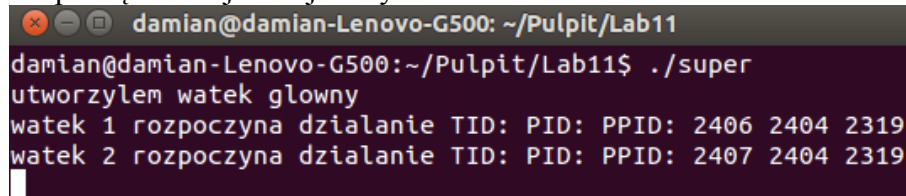
int main(void)
{
    init ();
    pthread_t watekglowny;
    pthread_create(&watekglowny,NULL,mainwatek,NULL);
    pthread_join (watekglowny,NULL);
    pthread_join(watek1,NULL);
    pthread_join(watek2,NULL);
    printf("zakonczone proces glowny\n");
}

```

Zmienna bufor2 powstała w celu wypisania tego co odczytał drugi wątek.

Wyniki uruchomienia

Na początek inicjalizacja i wywołanie:

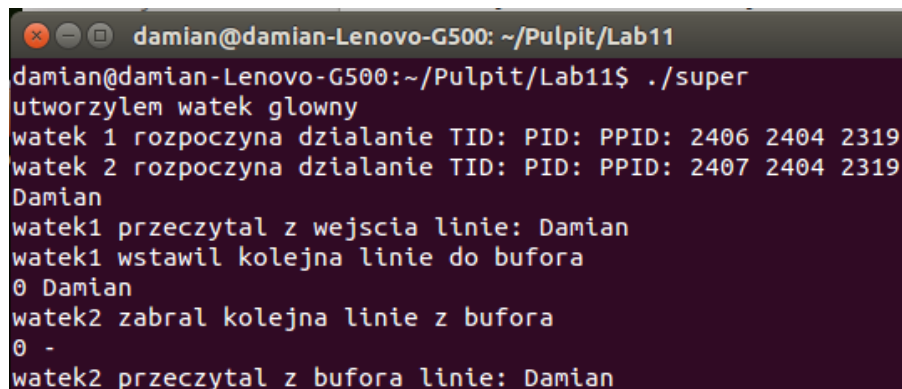


```

damian@damian-Lenovo-G500: ~/Pulpit/Lab11
damian@damian-Lenovo-G500:~/Pulpit/Lab11$ ./super
utworzylem watek glowny
watek 1 rozpoczyna dzialanie TID: PID: PPID: 2406 2404 2319
watek 2 rozpoczyna dzialanie TID: PID: PPID: 2407 2404 2319

```

Wpisujemy słowa:



```

damian@damian-Lenovo-G500: ~/Pulpit/Lab11
damian@damian-Lenovo-G500:~/Pulpit/Lab11$ ./super
utworzylem watek glowny
watek 1 rozpoczyna dzialanie TID: PID: PPID: 2406 2404 2319
watek 2 rozpoczyna dzialanie TID: PID: PPID: 2407 2404 2319
Damian
watek1 przeczytał z wejscia linie: Damian
watek1 wstawil kolejna linie do bufora
0 Damian
watek2 zabral kolejna linie z bufora
0 -
watek2 przeczytał z bufora linie: Damian

```

Może troszkę za dużo wypisywania, ale zawsze chciałem wiedzieć, w którym fragmencie programu jestem.

```
damian@damian-Lenovo-G500: ~/Pulpit/Lab11
damian@damian-Lenovo-G500:~/Pulpit/Lab11$ ./super
utworzyłem watek glowny
watek 1 rozpoczyna działanie TID: PID: PPID: 2406 2404 2319
watek 2 rozpoczyna działanie TID: PID: PPID: 2407 2404 2319
Damian
watek1 przeczytał z wejścia linie: Damian
watek1 wstawił kolejną linie do bufora
0 Damian
watek2 zabrał kolejną linie z bufora
0 -
watek2 przeczytał z bufora linie: Damian
Krata
watek1 przeczytał z wejścia linie: Krata
watek1 wstawił kolejną linie do bufora
0 Krata
watek2 zabrał kolejną linie z bufora
0 -
watek2 przeczytał z bufora linie: Krata
```

Wnioski:

Mutexy działają prawie identycznie jak semaforey, ale są w moim odczuciu dużo łatwiejsze. Służą do tego samego, a ich obsługa jest bardziej sprecyzowana. Powyższe ćwiczenie nauczyło mnie w pewnym stopniu posługiwać się mechanizmami synchronizacji.