Programowanie współbieżne

Laboratorium 6: Kolejki POSIX

Kamil Zdeb, nr albumu: 235871

Piątek TN 9:15-12:15

08.05.2020

1 Wstęp

Programy miały być uruchamiane na komputerze z systemem Linux.

2 Realizacja zadań

Programy wykonałem na posiadanej na swoim komputerze dystrybucji Linux Manjaro przy użyciu edytora *Sublime Text*. W celu sprawniejszej pracy użyłem narzędzia *make* ułatwiającego kompilację plików.

2.1 Zadanie 1: Problem producenta i konsumenta

Zadanie to składa się z trzech programów. Pierwszy, czyli init tworzy kolejkę POSIX, kolejny obsługuje konsumenta, a trzeci producenta. Program init wywołuje się bezparametrowo. Program konsumenta wywołuje się z jednym parametrem(liczba odbieranych towarów), a program producenta z dwoma parametrami(identyfikator producenta, liczba produkowanych towarów).

2.1.1 Kod programu

```
#include <stdio.h>
#include <mqueue.h>
#include <stdlib.h>
#include <unistd.h>
#include "common.h"
#define  MQ_NAME "/Kolejka"

main(int argc, char *argv[]) {
    mq_unlink(MQ_NAME);
    int i, res, num = 0;
    unsigned int prior;
    mqd_t mq;
    struct mq_attr attr;
    prior = 10;
    attr.mq_msgsize = sizeof(ms_type);
    attr.mq_maxmsg = 4;
    attr.mq_flags = 0;
    mq = mq_open(MQ_NAME , O_RDWR | O_CREAT , 0660, &attr );
    if( mq == -1 ) {
        perror("Kolejka ");
        exit(0);
    }
    printf("Kolejka: %s otwarta, mq: %d\n", MQ_NAME, mq);
    mq_close(mq);
}
```

Program konsumenta odbiera dane z kolejki za pomocą funkcji mq_receive.

```

#include <stdio.h>
#include <mqueue.h>
#include <stdlib.h>
```

```c
#include <unistd.h>
#include "common.h"
#define  MQ_NAME "/Kolejka"

int main(int argc, char *argv[]) {
 int i, res;
 unsigned int prior;
 mqd_t  mq;
 struct mq_attr attr;
 prior = 10;
 int steps=atoi(argv[1]);

 ms_type consumer;
 mq=mq_open(MQ_NAME , O_RDWR , 0660, &attr );
 if( mq == -1 ) {
  perror("Kolejka ");
  exit(0);
 }
 printf("Kolejka: %s otwarta, mq: %d\n", MQ_NAME,mq);
 for(i=0; i < steps ;i++) {

    res = mq_receive(mq,(char *)&consumer,sizeof(consumer),&prior);
    if (res == -1 )  perror("Blad odczytu z mq");
    else                printf("Odebrano: %s\n", consumer.text);
    sleep(1);
  }
  mq_close(mq);
}
```

Program producenta wysyła dane do kolejki POSIX za pomocą funkcji mq_send.

```c
#include <stdio.h>
#include <mqueue.h>
#include <stdlib.h>
#include <unistd.h>
#include "common.h"
#define  MQ_NAME "/Kolejka"
#define PROD 1

 int main(int argc, char *argv[]) {
   int i, res, num=0,steps;
   unsigned int prior;
   mqd_t  mq;
   struct mq_attr attr;
   prior = 10;
   if(argc < 2) { printf("Uzycie: mq_send numer\n"); exit(0); }
   num = atoi(argv[1]);
   steps=atoi(argv[2]);
   ms_type producer;
   // Otworzenie kolejki komunikatow ----------------

   mq=mq_open(MQ_NAME , O_RDWR , 0660, &attr );
   if( mq == -1 ) {  perror("Kolejka "); exit(0);  }
   printf("Kolejka: %s otwarta, mq: %d\n", MQ_NAME,mq);
   for(i=0; i < steps ;i++) {
     sprintf(producer.text,"Producent %d komunikat %d",num,i);
     producer.pnr = num;
     producer.type = PROD;
```

```
28      res = mq_send(mq,(char *)&producer,sizeof(producer.text),prior
        );
29      if (res == -1 ) { perror("Blad zapisu do mq");
30       continue;
31      }
32      printf("Producent %d Komunikat %d wyslany\n",num,i);
33     sleep(1);
34   }
35   mq_close(mq);
36 }
```

Wszystkie pliki korzystają z pliku nagłówkowego common.h.

```
1 #define SIZE 512
2
3 typedef struct {
4  int type; /* typ procesu: 1 PROD, 2 KONS */
5  int pnr ; /* numer procesu */
6  char text[SIZE]; /* tekst komunikatu */
7  } ms_type;
```

2.2 Zadanie 2: Znajdowanie liczb pierwszych - wersja równoważąca obciążenia

Program ten stworzyłem modyfikując analogiczny program z poprzednich zajęć, który działał na łączach. Proces potomny poza procesami potomnymi tworzy również wątek zapiszący do kolejki wejściowej kolejne podprzedziały zawarte w strukturach. Na końcu zapisuje również specjalne struktury zawierające przedział 0-0, którego odczytanie jest warunkiem stopu procesów szukających liczb pierwszych. Program tworzy 4 procesy szukające liczb pierwszych, ponieważ mój procesor ma 4 rdzenie. Program przyjmuje z terminala jako parametry początek przedziału, koniec przedziału oraz liczbę tworzonych podprzedziałów.

Rysunek 1: Efekt wykonania się programu

```
[kamil@kamil-pc lab6]$ ./zad2 2 1000 10
Kolejka: /KolejkaWejsciowa otwarta, mq: 3
Kolejka: /KolejkaWyjsciowa otwarta, mq: 4
Kolejka: /KolejkaWejsciowa otwarta, mq: 3
Kolejka: /KolejkaWyjsciowa otwarta, mq: 4
Kolejka: /KolejkaWejsciowa otwarta, mq: 3
Kolejka: /KolejkaWyjsciowa otwarta, mq: 4
Przedział'2-101, znaleziono 29 liczb pierwszych
Przedział'101-200, znaleziono 23 liczb pierwszych
Przedział'299-398, znaleziono 17 liczb pierwszych
Przedział'398-497, znaleziono 16 liczb pierwszych
Przedział'497-596, znaleziono 15 liczb pierwszych
Kolejka: /KolejkaWejsciowa otwarta, mq: 3
Kolejka: /KolejkaWyjsciowa otwarta, mq: 4
Koncze
Kolejka: /KolejkaWejsciowa otwarta, mq: 3
Kolejka: /KolejkaWyjsciowa otwarta, mq: 4
Koncze
Koncze
Przedział'596-695, znaleziono 17 liczb pierwszych
Przedział'695-794, znaleziono 13 liczb pierwszych
Przedział'794-893, znaleziono 17 liczb pierwszych
Przedział'893-992, znaleziono 14 liczb pierwszych
Przedział'200-299, znaleziono 17 liczb pierwszych
Koncze
Time elapsed: 0.002625 seconds
Number of primes: 178
```

2.2.1 Kod programu

Poniższy program tworzy kolejki wejściową i wyjściową, następnie je otwiera. Kolejnym krokiem jest utworzenie procesu piszącego do kolejki wejściowej, który zapisuje tam struktury zawierające przedział do policzenia. Poza tym tworzone są 4 procesy potomne odpowiadające za szukanie liczb pierwszych w podprzedziałach. Proces macierzysty czeka na zakończenie procesów potomnych oraz czyta z kolejki wyjściowej zliczając ile liczb pierwszych zostało znalezionych. Gdy skończy to robić zwraca liczbę znalezionych liczb pierwszych oraz czas, w którym zostały znalezione.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <math.h>
#include <tgmath.h>
#include <time.h>
#include <errno.h>
#include <fcntl.h>
#include <mqueue.h>
#define  MQ_NAME_ENTER "/KolejkaWejsciowa"
```

```c
#define  MQ_NAME_EXIT "/KolejkaWyjsciowa"


void main(int argc, char * argv[]){
  typedef struct {
    int nr; // numer przedzialu i
    int pocz; // poczatek zakresu obliczen Zd(i)
    int kon; // koniec zakresu obliczen Zg(i)
    int liczb; // ile liczb pierwszych w przedziale
  } result;
  struct timespec start, finish;
  double time_elapsed;
  clock_gettime(CLOCK_MONOTONIC, &start);
  int status;
  int lower_bound = atoi(argv[1]);
  int upper_bound = atoi(argv[2]);
  int bounds = atoi(argv[3]);
  int processes = 4;
  int pids[processes];
    int number_of_primes = 0;
  int bound = (upper_bound-lower_bound)/bounds;
  int writer;

  int res;
  mq_unlink(MQ_NAME_ENTER);
  mq_unlink(MQ_NAME_EXIT);
  mqd_t mq, mq2;
    struct mq_attr attr;
    struct mq_attr atrib;
    unsigned int prior = 10;
    attr.mq_msgsize = sizeof(result);
    attr.mq_maxmsg = 10;
    attr.mq_flags = 0;
    atrib.mq_msgsize = sizeof(result);
    atrib.mq_maxmsg = 10;
    atrib.mq_flags = 0;
    mq = mq_open(MQ_NAME_ENTER , O_RDWR | O_CREAT , 0660, &attr );
    if( mq == -1 ) {
        perror("Kolejka ");
        exit(0);
    }
    printf("Kolejka: %s otwarta, mq: %d\n", MQ_NAME_ENTER, mq);

    mq2 = mq_open(MQ_NAME_EXIT , O_RDWR | O_CREAT , 0660, &atrib);
    if( mq2 == -1 ) {
        perror("Kolejka ");
        exit(0);
    }
    printf("Kolejka: %s otwarta, mq: %d\n", MQ_NAME_EXIT, mq2);


  if((writer= fork()) == 0){
    for(int i=0 ; i<bounds; i++){
      int lb = lower_bound+i*bound;
            int ub = lower_bound+(i+1)*bound;
            result my_res = {i, lb, ub, 0};
      res = mq_send(mq,(char *)&my_res, sizeof(my_res),prior);
```

```
      //printf("Wys ano przedzia  (%d-%d)\n", lb, ub);
    }
    for (int i=0; i<processes; i++){
      result my_res = {0,0,0,0};
      res = mq_send(mq,(char *)&my_res, sizeof(my_res),prior);
      //printf("Wys ano przedzia  (%d-%d)\n", 0, 0);
    }
    exit(0);

  }
  for(int i=0; i<processes; i++){
    if((pids[i] = fork()) == 0) { /* Proces potomny ---*/

            execl("./licz","licz",NULL);
    }
  }
 /* Proces macierzysty */

  int bytes_read, counter=0;
  result wynik;
  do{
  res = mq_receive(mq2,(char *)&wynik,sizeof(wynik),&prior);
    if (res == -1 )  perror("Blad odczytu z mq");
  printf("Przedzia '%d-%d, znaleziono %d liczb pierwszych\n",
    wynik.pocz, wynik.kon, wynik.liczb);
  counter++;
  number_of_primes+=wynik.liczb;
  //printf("%d\n", bytes_read);
  }while(counter<bounds);
  for(int i=0; i<processes; i++){
    pids[i] += wait(&status);
        WEXITSTATUS(status);
  }
  writer += wait(&status);
    WEXITSTATUS(status);
  clock_gettime(CLOCK_MONOTONIC, &finish);
  time_elapsed = (finish.tv_sec - start.tv_sec);
  time_elapsed += (finish.tv_nsec - start.tv_nsec) / 1000000000.0;
  printf("Time elapsed: %f seconds\n", time_elapsed);
    printf("Number of primes: %d \n", number_of_primes);
    mq_close(mq);
    mq_close(mq2);
}

}
```

Poniższy fragment kodu zawarty w pliku *licz.c* odpowiada za otwarcie kolejki wejściowej i wyjściowej. Następnie pobiera dane z kolejki wejściowej, szuka liczb pierwszych dla podanego podprzedziału, a następnie zapisuje strukturę do kolejki wyjściowej. W momencie odczytania przedziału 0-0 program zamyka kolejki i kończy działanie. Próbowałem skorzystać z atrybutu przechowującego dane o liczbie wiadomości w kolejce, ale przy próbie ich odczytania znajdowały się tam śmieci i program nigdy nie osiągał swojego warunku stopu.

```
 #include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

```c
#include <math.h>
#include <tgmath.h>
#include <stdbool.h>
#include <time.h>
#include <errno.h>
#include <fcntl.h>
#include <mqueue.h>
#define  MQ_NAME_ENTER "/KolejkaWejsciowa"
#define  MQ_NAME_EXIT "/KolejkaWyjsciowa"

int prime(int n);

int main(int argc, char * argv[]){
    typedef struct {
        int nr; // numer przedzialu i
        int pocz; // poczatek zakresu obliczen Zd(i)
        int kon; // koniec zakresu obliczen Zg(i)
        int liczb; // ile liczb pierwszych w przedziale
    } result;

    struct mq_attr attr, attr2;
    mqd_t mq, mq2;
    unsigned int prior = 10;
    int number_of_primes = 0;
    int res;

    mq = mq_open(MQ_NAME_ENTER , O_RDWR , 0660, &attr);
    if( mq == -1 ) {
        perror("Kolejka ");
        exit(0);
    }
    printf("Kolejka: %s otwarta, mq: %d\n", MQ_NAME_ENTER, mq);
    mq2 = mq_open(MQ_NAME_EXIT , O_RDWR , 0660, &attr);
    if( mq2 == -1 ) {
        perror("Kolejka ");
        exit(0);
    }
    printf("Kolejka: %s otwarta, mq: %d\n", MQ_NAME_EXIT, mq2);
    result my_res;
    mq_getattr(mq, &attr);
    while(1){
        // mq_getattr(mq, &attr);
        // long messages = attr.mq_curmsgs;
        // if (messages==0){
        //     exit(0);
        // }
        // printf("Wiadomo ci:%d\n", messages);
        number_of_primes = 0;
        res = mq_receive(mq,(char *)&my_res,sizeof(my_res),&prior);
        int lower_bound = my_res.pocz;
        int upper_bound = my_res.kon;
        //printf("Odebrano przedzia  (%d-%d)\n", lower_bound,
    upper_bound);
        if(lower_bound==0&&upper_bound==0){
            mq_close(mq);
            mq_close(mq2);
            printf("Koncze\n");
```

```
60              exit(0);
61
62          }
63          for(int i=lower_bound; i<upper_bound; i++){
64              number_of_primes += prime(i);
65          }
66          my_res.liczb = number_of_primes;
67          res = mq_send(mq2,(char *)&my_res, sizeof(my_res),prior);
68      }
69
70  }
71
72  int prime(int n){
73      for(int j=2; j*j<n; j++){
74              if(n%j==0) return(0);
75          }
76      return(1);
77  }
78
79  }
```