Programowanie współbieżne

Laboratorium 5: Łącza nazwane i nienazwane

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: Znajdowanie liczb pierwszych - łącza nienazwane

Program przyjmuje z terminala trzy argumenty: początek przedziału, koniec przedziału oraz liczba procesów potomnych. Każdy proces potomny zlicza liczby pierwsze w danym przedziale, a następnie zapisuje strukturę z danymi do łącza nienazwanego. Proces macierzysty odczytuje dane z łącza nienazwanego, sumuje liczbę znalezionych liczb pierwszych i wyświetla ją na ekranie wraz z czasem działania programu. Efekt wykonania się programu znajduje się w poniższym zrzucie ekranu.

Rysunek 1: Efekt wykonania się programu

```
[kamil@kamil-pc lab5]$ ./zad1 2 1000000 10
Przedział'2-100001, znaleziono 9657 liczb pierwszych
Przedział'299999-399998, znaleziono 7877 liczb pierwszych
Przedział'200000-299999, znaleziono 8028 liczb pierwszych
Przedział'100001-200000, znaleziono 8413 liczb pierwszych
Przedział'399998-499997, znaleziono 7689 liczb pierwszych
Przedział'599996-699995, znaleziono 7454 liczb pierwszych
Przedział'799994-899993, znaleziono 7331 liczb pierwszych
Przedział'699995-799994, znaleziono 7416 liczb pierwszych
Przedział'499997-599996, znaleziono 7570 liczb pierwszych
Przedział'899993-999992, znaleziono 7231 liczb pierwszych
Time elapsed: 0.062514 seconds
Number of primes: 78666
```

2.1.1 Kod programu

```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>

int prime(int n);

void main(int argc, char * argv[]){
```

```
  struct result {
    int pocz;// pocz tek przedzialu
    int kon; // koniec przedzialu
    int ile; // Ile liczb w przedziale
  };
  struct timespec start, finish;
  double time_elapsed;
  clock_gettime(CLOCK_MONOTONIC, &start);
  int fd[2];
  int status;
  int lower_bound = atoi(argv[1]);
  int upper_bound = atoi(argv[2]);
  int processes = atoi(argv[3]);
  int pids[processes];
    int number_of_primes = 0;
  int bound = (upper_bound-lower_bound)/processes;
  pipe(fd);
  for(int i=0; i<processes; i++){
    if((pids[i] = fork()) == 0) { /* Proces potomny ---*/
      close(fd[0]);
            int lb = lower_bound+i*bound;
            int ub = lower_bound+(i+1)*bound;
            int primes_found = 0;
            for(int j = lb; j<ub;j++){
              primes_found+=prime(j);
            }
            struct result my_res = {lb, ub, primes_found};
            write(fd[1], &my_res, sizeof(my_res));
            close(fd[1]);
            exit(0);
    }
  }
  close(fd[1]);
  int counter = 0;
  struct result wynik;
  for(int i=0; i<processes; i++){
    pids[i] += wait(&status);
        WEXITSTATUS(status);
  }
  while(counter<processes){
    read(fd[0], &wynik, sizeof(wynik));
    printf("Przedzia '%d-%d, znaleziono %d liczb pierwszych\n",
    wynik.pocz, wynik.kon, wynik.ile);
    number_of_primes += wynik.ile;
    counter++;
  }
  close(fd[0]);
  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);
}

int prime(int n){
    for(int j=2; j*j<n; j++){
            if(n%j==0) return(0);
```

```
69          }
70      return(1);
71  }
```

2.2 Zadanie 2: Znajdowanie liczb pierwszych - kolejka FIFO

Drugie zadanie polegało na rozwiązaniu tego samego problemu z wykorzystaniem łącz nazwanych(kolejki FIFO). Zadanie to było w swojej strukturze podobne do zadania z poprzednich zajęć, które operowało na plikach. Program przyjmuje ten sam zestaw parametrów co poprzednie zadanie, czyli początek przedziału, koniec przedziału oraz liczbę procesów potomnych. Program tworzy procesy potomne za pomocą funkcji fork oraz execl. Wykorzystuje program potomny zawarty w pliku licz.c.

Rysunek 2: Efekt wykonania się programu

```
[kamil@kamil-pc lab5]$ ./zad2 2 1000000 10
Przedział'2-100001, znaleziono 9657 liczb pierwszych
Przedział'100001-200000, znaleziono 8413 liczb pierwszych
Przedział'200000-299999, znaleziono 8028 liczb pierwszych
Przedział'299999-399998, znaleziono 7877 liczb pierwszych
Przedział'899993-999992, znaleziono 7231 liczb pierwszych
Przedział'399998-499997, znaleziono 7689 liczb pierwszych
Przedział'699995-799994, znaleziono 7416 liczb pierwszych
Przedział'499997-599996, znaleziono 7570 liczb pierwszych
Przedział'599996-699995, znaleziono 7454 liczb pierwszych
Przedział'799994-899993, znaleziono 7331 liczb pierwszych
Time elapsed: 0.079542 seconds
Number of primes: 78666
```

2.2.1 Kod programu

Poniższy kod prezentuje program tworzący procesy potomne, tworzący i otwierający łącze nazwane oraz zliczający liczbę znalezionych przez procesy potomne liczb pierwszych. Kod tego programu znajduje się w pliku *zad2.c*.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <math.h>
5  #include <tgmath.h>
6  #include <time.h>
7  #include <errno.h>
8  #include <fcntl.h>
9
10 void main(int argc, char * argv[]){
11   struct result {
12     int pocz;// pocz tek przedzialu
13     int kon; // koniec przedzialu
14     int ile; // Ile liczb w przedziale
15   };
16   struct timespec start, finish;
17   double time_elapsed;
18   clock_gettime(CLOCK_MONOTONIC, &start);
```

```
  int status;
  int lower_bound = atoi(argv[1]);
  int upper_bound = atoi(argv[2]);
  int processes = atoi(argv[3]);
  int pids[processes];
    int number_of_primes = 0;
  int bound = (upper_bound-lower_bound)/processes;
  int fd;
  unlink("FIFO");
  if(mkfifo("FIFO",0666) < 0) {
    perror("mkfifo");
  }
  fd = open("FIFO", O_RDWR, S_IRWXU);
    if(fd<0) {
    perror("open");
    exit(0);
  }
  for(int i=0; i<processes; i++){
    if((pids[i] = fork()) == 0) { /* Proces potomny ---*/
            int lb = lower_bound+i*bound;
            int ub = lower_bound+(i+1)*bound;
            //printf("Dolny przedzia : %d \t G rny przedzia : %d
    \n",lb,ub);
      char buffer[50];
            char buffer2[50];
            sprintf(buffer, "%d ", lb);
            sprintf(buffer2, "%d ", ub);
            execl("./licz","licz", buffer, buffer2, NULL);

    }
  }

 /* Proces macierzysty */
  for(int i=0; i<processes; i++){
    pids[i] += wait(&status);
        WEXITSTATUS(status);
  }

  int bytes_read, counter=0;
  while (counter<processes){
  struct result wynik;
  bytes_read = read(fd,&wynik,sizeof(wynik));
  printf("Przedzia '%d-%d, znaleziono %d liczb pierwszych\n",
    wynik.pocz, wynik.kon, wynik.ile);
  counter++;
  number_of_primes+=wynik.ile;
  }
  close(fd);
  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);
}
```

Poniższy kod zawiera uruchamiany w programie *zad2* proces potomny, który odpowiada za znalezienie liczb pierwszych w podanym przedziale oraz zapisanie

znalezionych informacji w stukturze, którą następnie przekazuje do łącza nazwanego, z którego proces macierzysty następnie odczytuje dane. Kod źródłowy jest dostępny także w zawartości pliku *licz.c*.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <math.h>
#include <tgmath.h>
#include <stdbool.h>
#include <time.h>
#include <errno.h>
#include <fcntl.h>

int prime(int n);

int main(int argc, char * argv[]){
    struct result {
        int pocz;// pocz tek przedzialu
        int kon; // koniec przedzialu
        int ile; // Ile liczb w przedziale
    };
    int fd, res;
    int lower_bound = atoi(argv[1]);
    int upper_bound = atoi(argv[2]);
    int number_of_primes = 0;
    //printf("Dolny przedzia : %d \t G rny przedzia : %d\n",
    lower_bound, upper_bound);
    for(int i=lower_bound; i<upper_bound; i++){
        number_of_primes += prime(i);
    }
    struct result my_res = {lower_bound, upper_bound,
    number_of_primes};

    fd = open("FIFO", O_RDWR);
    if(fd<0) {
        perror("open");
        exit(0);
    }
    write(fd, &my_res, sizeof(my_res));
    close(fd);
    return 0;
}

int prime(int n){
    for(int j=2; j*j<n; j++){
            if(n%j==0) return(0);
        }
    return(1);
}
```

2.3 Zadanie 3: Znajdowanie liczb pierwszych - wersja równoważąca obciążenia

Trzeci program różni się od poprzedniego tym, że w swoim działaniu wykorzystuje dwa łącza nazwane - wejściowe i wyjściowe. Proces potomny poza procesami potomnymi tworzy również wątek zapiszący do łącza nazwanego 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 3: Efekt wykonania się programu

```
[kamil@kamil-pc lab5]$ ./zad3 2 1000000 25
Przedział'2-40001, znaleziono 4249 liczb pierwszych
Przedział'40001-80000, znaleziono 3648 liczb pierwszych
Przedział'80000-119999, znaleziono 3472 liczb pierwszych
Przedział'119999-159998, znaleziono 3392 liczb pierwszych
Przedział'159998-199997, znaleziono 3308 liczb pierwszych
Przedział'239996-279995, znaleziono 3218 liczb pierwszych
Przedział'199997-239996, znaleziono 3244 liczb pierwszych
Przedział'279995-319994, znaleziono 3180 liczb pierwszych
Przedział'319994-359993, znaleziono 3155 liczb pierwszych
Przedział'359993-399992, znaleziono 3109 liczb pierwszych
Przedział'399992-439991, znaleziono 3086 liczb pierwszych
Przedział'439991-479990, znaleziono 3069 liczb pierwszych
Przedział'479990-519989, znaleziono 3057 liczb pierwszych
Przedział'519989-559988, znaleziono 3016 liczb pierwszych
Przedział'559988-599987, znaleziono 3030 liczb pierwszych
Przedział'599987-639986, znaleziono 2979 liczb pierwszych
Przedział'639986-679985, znaleziono 2992 liczb pierwszych
Przedział'679985-719984, znaleziono 2970 liczb pierwszych
Przedział'719984-759983, znaleziono 2954 liczb pierwszych
Przedział'759983-799982, znaleziono 2974 liczb pierwszych
Przedział'799982-839981, znaleziono 2942 liczb pierwszych
Przedział'839981-879980, znaleziono 2938 liczb pierwszych
Przedział'879980-919979, znaleziono 2913 liczb pierwszych
Przedział'919979-959978, znaleziono 2888 liczb pierwszych
Przedział'959978-999977, znaleziono 2881 liczb pierwszych
Time elapsed: 0.067944 seconds
Number of primes: 78664
```

2.3.1 Kod programu

Poniższy program tworzy łącza nazwane wejściowe i wyjściowe, następnie je otwiera. Kolejnym krokiem jest utworzenie procesu piszącego do łącza wejściowego, 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 łącza wyjściowego 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>

void main(int argc, char * argv[]){
  struct result {
    int pocz;// pocz tek przedzialu
    int kon; // koniec przedzialu
    int ile; // Ile liczb w przedziale
  };
  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 fd_enter, fd_exit;
  int writer;
  unlink("FIFO_wejscie");
  if(mkfifo("FIFO_wejscie",0666) < 0) {
    perror("mkfifo");
  }
  unlink("FIFO_wyjscie");
  if(mkfifo("FIFO_wyjscie",0666) < 0) {
    perror("mkfifo");
  }
  fd_exit = open("FIFO_wyjscie", O_RDWR, S_IRWXU);
    if(fd_exit<0) {
    perror("open");
    exit(0);
  }
  fd_enter = open("FIFO_wejscie", O_RDWR, S_IRWXU);
  if(fd_enter<0) {
    perror("open");
    exit(0);
  }
  if((writer= fork()) == 0){
    for(int i=0 ; i<bounds; i++){
      int lb = lower_bound+i*bound;
            int ub = lower_bound+(i+1)*bound;
            struct result res = {lb, ub, 0};
      write(fd_enter, &res, sizeof(res));
    }
    for(int i = 0; i<processes; i++){
      struct result res = {0,0,0};
      write(fd_enter, &res, sizeof(res));
```

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

            execl("./licz2","licz2",NULL);
    }
  }

 /* Proces macierzysty */
  for(int i=0; i<processes; i++){
    pids[i] += wait(&status);
        WEXITSTATUS(status);
  }
  writer += wait(&status);
    WEXITSTATUS(status);
  int bytes_read, counter=0;
  struct result wynik;
  do{
  bytes_read = read(fd_exit,&wynik,sizeof(wynik));
  if (bytes_read == 0) break;
  printf("Przedzia '%d-%d, znaleziono %d liczb pierwszych\n",
    wynik.pocz, wynik.kon, wynik.ile);
  counter++;
  number_of_primes+=wynik.ile;
  //printf("%d\n", bytes_read);
  }while(counter<bounds);
  close(fd_exit);
  close(fd_enter);
  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);
}
```

Poniższy fragment kodu zawarty w pliku *licz2.c* odpowiada za otwarcie łącza nazwanego wejściowego oraz wyjściowego. Następnie pobiera dane z łącza wejściowego, szuka liczb pierwszych dla podanego podprzedziału, a następnie zapisuje strukturę do łącza wyjściowego. W momencie odczytania przedziału 0-0 program zamyka łącza i kończy działanie.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <math.h>
#include <tgmath.h>
#include <stdbool.h>
#include <time.h>
#include <errno.h>
#include <fcntl.h>

int prime(int n);

int main(int argc, char * argv[]){
    struct result {
```

```
15          int pocz;// pocz tek przedzialu
16          int kon; // koniec przedzialu
17          int ile; // Ile liczb w przedziale
18      };
19      int fd_enter, fd_exit;
20      int number_of_primes = 0;
21
22
23      fd_enter = open("FIFO_wejscie", O_RDWR);
24      if(fd_enter<0) {
25          perror("open");
26          exit(0);
27      }
28      fd_exit = open("FIFO_wyjscie", O_RDWR);
29      if(fd_exit<0) {
30          perror("open");
31          exit(0);
32      }
33      struct result my_res;
34      int bytes_read;
35      while(1){
36          number_of_primes = 0;
37          read(fd_enter, &my_res, sizeof(my_res));
38          int lower_bound = my_res.pocz;
39          int upper_bound = my_res.kon;
40          if(lower_bound==0&&upper_bound==0){
41              close(fd_exit);
42              close(fd_enter);
43              return 0;
44          }
45          for(int i=lower_bound; i<upper_bound; i++){
46              number_of_primes += prime(i);
47          }
48          my_res.ile = number_of_primes;
49          write(fd_exit, &my_res, sizeof(my_res));
50      }
51
52  }
53
54  int prime(int n){
55      for(int j=2; j*j<n; j++){
56              if(n%j==0) return(0);
57          }
58      return(1);
59  }
```